



НТО

МАТЕРИАЛЫ ЗАДАНИЙ

Всероссийской междисциплинарной олимпиады школьников

«Национальная технологическая олимпиада»

по профилю

«Беспилотные авиационные системы»

2023/24 учебный год

<http://ntcontest.ru>

УДК 373.5.016:629.73.02
ББК 74.263.0
Б23

Авторы:

Д.С. Васильев, М.А. Григулецкий, К.Д. Кириченко, Е.С. Кодикова, Н.Ю. Кузнецов,
К.С. Лельков, М.Б. Прохоров, А.Н. Ридли, Д. А. Сурков, Д.В. Ульянов, А.Н. Ушаков,
Т.С. Хорев

Б23 Всероссийская междисциплинарная олимпиада школьников 8-11 класса
«Национальная технологическая олимпиада». Учебно-методическое пособие
Том 5 **Беспилотные авиационные системы**
—М.: ООО «ВАШ ФОРМАТ», 2024. —170 с.

ISBN 978-5-00147-595-8

Данное пособие разработано коллективом авторов на основе опыта проведения всероссийской междисциплинарной олимпиады школьников 8-11 класса «Национальная технологическая олимпиада» в 2023/24 учебном году, а также многолетнего опыта проведения инженерных соревнований для школьников. В пособии собраны основные материалы, необходимые как для подготовки к олимпиаде так и для углубления знаний и приобретения навыков решения инженерных задач.

В издании приведены варианты заданий по профилю Национальной технологической олимпиады за 2023/24 учебный год с ответами, подробными решениями и комментариями. Пособие адресовано учащимся 8–11 классов, абитуриентам, школьным учителям, наставникам и преподавателям учреждений дополнительного образования, центров молодежного и инновационного творчества и детских технопарков.

Методические материалы также могут быть полезны студентам и преподавателям направлений, относящихся к группам:

09.00.00 Информатика и вычислительная техника

11.00.00 Электроника, радиотехника и системы связи

12.00.00 Фотоника, приборостроение, оптические и биотехнические системы и технологии

24.00.00 Авиационная и ракетно-космическая техника

25.00.00 Аэронавигация и эксплуатация авиационной и ракетно-космической техники

27.00.00 Управление в технических системах

ISBN 978-5-00147-595-8

УДК 373.5.016:629.73.02
ББК 74.263.0



9 785001 475958 >

Оглавление

1 Введение	5
2 Беспилотные авиационные системы	17
I Работа наставника НТО на первом отборочном этапе	19
II Первый отборочный этап	20
II.1 Предметный тур. Информатика и программирование	20
II.1.1 Первая волна. Задачи 8–11 класса	20
II.1.2 Вторая волна. Задачи 8–11 класса	31
II.1.3 Третья волна. Задачи 8–11 класса	41
II.2 Предметный тур. Физика	51
II.2.1 Первая волна. Задачи 8–9 класса	51
II.2.2 Первая волна. Задачи 10–11 класса	56
II.2.3 Вторая волна. Задачи 8–9 класса	61
II.2.4 Вторая волна. Задачи 10–11 класса	66
II.2.5 Третья волна. Задачи 8–9 класса	71
II.2.6 Третья волна. Задачи 10–11 класса	77
II.3 Инженерный тур	83
III Работа наставника НТО на втором отборочном этапе	92
IV Второй отборочный этап	93
V Работа наставника НТО при подготовке к заключитель-	

ному этапу	111
VI Заключениеый этап	112
VI.1 Предметный тур	112
VI.1.1 Информатика и программирование. 8–11 классы	112
VI.1.2 Физика. 8–9 классы	123
VI.1.3 Физика. 10–11 классы	127
VI.2 Инженерный тур	135
VI.2.1 Общая информация	135
VI.2.2 Легенда задачи	135
VI.2.3 Требования к команде и компетенциям участников	135
VI.2.4 Оборудование и программное обеспечение	135
VI.2.5 Задачи	136
VII Критерии определения победителей и призеров	166
VIII Работа наставника после НТО	168

Введение

Национальная технологическая олимпиада

Всероссийская междисциплинарная олимпиада школьников «Национальная технологическая олимпиада» (далее — НТО) проводится в соответствии с распоряжением Правительства Российской Федерации от 10.02.2022 № 211-р при координации Министерства науки и высшего образования Российской Федерации и при содействии Министерства просвещения Российской Федерации, Министерства цифрового развития, связи и массовых коммуникаций Российской Федерации, Министерства промышленности и торговли Российской Федерации, Ассоциации участников технологических кружков, Агентства стратегических инициатив по продвижению новых проектов, АНО «Россия — страна возможностей», АНО «Платформа Национальной технологической инициативы».

Проектное управление Олимпиадой осуществляет структурное подразделение Национального исследовательского университета «Высшая школа экономики» — Центр Национальной технологической олимпиады. Организационный комитет по подготовке и проведению Национальной технологической олимпиады возглавляют первый заместитель Руководителя Администрации Президента Российской Федерации С. В. Кириенко и заместитель Председателя Правительства Российской Федерации Д. Н. Чернышенко.

Всероссийская междисциплинарная олимпиада школьников 8–11 класса «Национальная технологическая олимпиада» — это командная инженерная Олимпиада, позволяющая школьникам работать в 41-м инженерном направлении. Она базируется на опыте Олимпиады Кружкового движения НТИ и проводится с 2015 года, а с 2016 года входит в перечень Российского совета олимпиад школьников и дает победителям и призерам льготы при поступлении в университеты.

Всего заявки на участие в девятом сезоне (2023–24 гг.) самых масштабных в России командных инженерных соревнованиях подали более 141 тысячи школьников и студентов из всех регионов страны и семи зарубежных государств: Азербайджана, Белоруссии, Казахстана, Киргизии, Молдовы, Узбекистана и Черногории. Общий охват олимпиады с 2015 года превысил 660 000 участников. <https://journal.kruzhok.org/tpost/pggs3bp7y1-tehnologicheskaya-podgotovka-inzhenernih>



НТО способствует формированию профессиональной траектории школьников, увлеченных научно-техническим творчеством:

- определить свой интерес в мире современных технологий;
- получить опыт решения комплексных инженерных задач;
- осознанно выбрать вуз для продолжения обучения и поступить в него на льготных условиях.

Кроме того, НТО позволяет каждому участнику познакомиться с перспективными направлениями технологического развития и ведущими экспертами, а также найти единомышленников.

Ценности НТО

Национальная технологическая олимпиада — командные инженерные соревнования для школьников и студентов. Особое пространство Олимпиады создают общие ценности и смыслы, которые предлагается разделять всем: участникам, организаторам, наставникам, экспертам.

Основа всей олимпиады — это современное технологическое образование как новый уклад жизни в современном мире. Этот уклад подразумевает доступность качественного образования для каждого заинтересованного человека, возможность постепенно и непрерывно учиться и развиваться, совместно создавать среду, в которой гуманитарное знание и новые технологии взаимно дополняют друг друга. Это идеал будущего общества. Участники Олимпиады уже сейчас попадают в такое будущее.

Как организаторы мы надеемся, что принципы, заложенные в основу НТО, станут общими принципами для всех, кто имеет отношение к Олимпиаде.

Решать прикладные задачи, нацеленные на умножение общественного блага

В соревнованиях и подготовке к ним мы адаптируем реальные задачи современной науки и производства к знаниям и навыкам, которые могут освоить школьники и студенты. Задачи имеют прикладное значение для людей и не оторваны от реальности. Мы стремимся к тому, чтобы участники понимали, для чего нужно решать такие задачи, кому в мире станет лучше, если они будут решаться системно и профессионально. Ценность Олимпиады заключается в том, что здесь можно попробовать себя в этом, и найти единомышленников для решения подобных задач в будущем.

Создавать, а не только потреблять

Создание новых решений мы ставим выше стремления потреблять уже созданное. Создание ценности для других ставим выше поиска личной выгоды. Это не значит, что нужно забыть о себе и самоотверженно посвятить всю свою жизнь делу технологического прогресса. Но творчество всегда приносит большую радость, чем потребление. Это относится и ко всей олимпиаде.

Олимпиада — это общее дело организаторов, партнеров и участников. Способность принимать проблемы олимпиады как свои и пытаться решить их ценнее для творческого человека, чем желание найти недостатки в работе других.

Работать в команде

Способность работать в команде — это не только эффективная стратегия действия в современном мире. Работа в команде не отрицает наличия свободной воли каждого конкретного участника, его значимости и права на собственное мнение. Но в сообществе мы стремимся достигнуть общей цели, опираясь на взаимное уважение всех участников, учитывая интересы и слабые и сильные стороны каждого.

Команды формируют целые сообщества, которые имеют сходные цели и ценности и могут очень многое, поскольку сильные горизонтальные связи помогают реализовывать самые дерзкие и амбициозные задачи. Это то, что нужно для технологического развития. Мы заняты построением такого сообщества и надеемся, что вы захотите стать его частью.

Осваивать и ответственно развивать новые технологии

Сообщество Национальной технологической олимпиады — часть Кружкового движения НТИ. Это прежде всего сообщество людей, увлеченных современными технологиями. Нас всех объединяет стремление разобраться в них, создать что-то новое и найти таких же увлеченных единомышленников.

Мы — часть сообщества технологических энтузиастов, и для нас границы возможностей технологий всегда подвижны. Именно поэтому просим не забывать об этике инженера и ученого, ответственности за свои изобретения перед людьми, которых это касается. Творя новое, не навреди!

Играть честно и пробовать себя

Мы признаем, что победа в соревнованиях важна и нужна. Но утверждаем, что для победы не все средства хороши и цель не является оправданием для грязной игры. Победа должна быть заслужена в рамках правил, единых для всех. Человек, который играет честно, не будет списывать, интриговать, подставлять других и заниматься прочей нездоровой конкуренцией.

Человек, который играет честно, — уважает себя, свою команду и соперников. Он принимает правила игры и в заданных рамках доказывает право на победу.

Мы бережем пространство Олимпиады как безопасное для всех участников. Это помогает искать себя, и при этом не бояться пробовать новые задачи, определять свой дальнейший путь, учиться на ошибках и каждый год становиться более сильным и подготовленным.

Быть человеком

Соревнования — это очень сложный и эмоционально насыщенный процесс. Что бы он приносил радость и пользу всем, мы призываем всех участников вести себя порядочно и думать не только о себе.

Вежливость, эмпатия и забота — вот что делает процесс комфортным и полезным для всех. Мы ценим уважение труда каждого человека и его позиции, бережное отношение к работе и жизни каждого. И просим отказаться от токсичной оценочной критики — она не решит ваши проблемы, а сделает хуже вам, другому и всей

Олимпиаде в целом.

Человек, который остается человеком, умеет признавать ошибки и отвечать за слова и дела перед другими. Здесь это ценят. Встав перед альтернативой между сиюминутной выгодой, капризом и общей целью соревнования — человек выберет последнее и поможет другим, организаторам и участникам, поддержать эту цель.

Важное замечание. Этот текст — живое выражение смыслов и ценностей Национальной технологической олимпиады. Он будет меняться вместе с развитием нашего сообщества. Авторы с благодарностью примут помощь от всех, кто чувствует сопричастность ценностям и готов включиться в их доработку.

Организационная структура НТО

НТО — межпредметная олимпиада. Спектр соревновательных направлений (профилей НТО) сформирован на основе актуального технологического пакета и связан с решением современных проблем в различных технологических отраслях. С полным перечнем направлений (профилей) можно ознакомиться на сайте НТО: <https://ntcontest.ru/tracks/nto-school/>.



Соревнования в рамках НТО проводятся по четырем направлениям:

1. НТО Junior для школьников (5–7 классы).
2. НТО школьников (8–11 классы).
3. НТО студентов.
4. Конкурс цифровых портфолио «Талант НТО».

В 2023/24 учебном году 28 профилей НТО включены в Перечень олимпиад школьников, утверждаемый Приказом Министерства науки и высшего образования Российской Федерации, а также в Перечень олимпиад и иных интеллектуальных и (или) творческих конкурсов, утверждаемый приказом Министерства просвещения Российской Федерации, что дает право победителям и призерам профилей НТО поступать в вузы страны без вступительных испытаний (БВИ), получить 100 баллов ЕГЭ или дополнительные 10 баллов за индивидуальные достижения. Преимущества при поступлении победителям и призерам НТО предлагают более 100 российских вузов.

НТО для старшеклассников проводится в три этапа:

- Первый отборочный этап — заочный индивидуальный. На данном этапе участникам предлагаются задачи по двум предметам, соответствующим тому или

иному профилю, а также задания, формирующие теоретические знания и представления по направлениям выбранных профилей.

- Второй отборочный этап — заочный командный. На данном этапе участникам предлагаются индивидуальные компетентностные и командные задачи, связанные с направлением выбранного профиля.
- Заключительный этап — очный командный. Этап представляет собой очные соревнования длительностью 5–6 дней, куда приезжают команды со всей страны, успешно справившиеся с двумя отборочными этапами, и решают комплексные прикладные инженерные задачи.

Профили НТО 2023/24 учебного года и соответствующий уровень РСОШ

Профили II уровня РСОШ

- Автоматизация бизнес-процессов
- Беспилотные авиационные системы
- Водные робототехнические системы
- Инженерные биологические системы
- Интеллектуальные робототехнические системы
- Нейротехнологии и когнитивные науки
- Технологии беспроводной связи

Профили III уровня РСОШ

- Автономные транспортные системы
- Анализ космических снимков и геопространственных данных
- Аэрокосмические системы
- Большие данные и машинное обучение
- Геномное редактирование
- Интеллектуальные энергетические системы
- Информационная безопасность
- Искусственный интеллект
- Летящая робототехника
- Наносистемы и наноинженерия
- Новые материалы
- Передовые производственные технологии
- Разработка компьютерных игр
- Спутниковые системы
- Технологии виртуальной реальности
- Технологии дополненной реальности
- Технологическое предпринимательство
- Умный город
- Фотоника
- Цифровые технологии в архитектуре
- Ядерные технологии

Профили без уровня РСОШ

- Научная медиакоммуникация
- Программная инженерия в финансовых технологиях
- Современная пищевая инженерия
- Технологическое мейкерство
- Урбанистика
- Цифровое производство в машиностроении
- Цифровой инжиниринг в строительстве
- Цифровые сенсорные системы

Новые профили без уровня РСОШ

- Инфохимия
- Квантовый инжиниринг
- Технологии компьютерного зрения и цифровые сервисы
- Цифровая гидрометеорология
- Цифровое месторождение

Обратите внимание, что в олимпиаде 2024/25 года список профилей, в т.ч. входящих в РСОШ, и уровни РСОШ — могут поменяться.

Участие в НТО может принять любой школьник, обучающийся в 8–11 классе. Чаще всего Олимпиада привлекает:

- учащихся технологических кружков, любители инженерных и робототехнических соревнований;
- олимпиадников, которым интересны межпредметные олимпиады;
- фанатов и адептов передовых технологий;
- школьников, участвующих в хакатонах, проектных конкурсах и школах;
- будущих предпринимателей, намеревающихся найти на Олимпиаде единомышленников для будущего стартапа;
- увлекающихся школьников, которые хотят видеть предмет шире учебника.

Познакомить школьников с НТО и ее направлениями, замотивировать принять участие в НТО можно с помощью специальных мероприятий: Урок НТО и Дни НТО. Как педагогу провести Урок НТО, или как в образовательном учреждении организовать День НТО можно познакомиться в методических рекомендациях на сайте НТО. Там же можно выбрать и скачать необходимые уроки и подборки материалов по направлениям <https://nti-lesson.ru/>.



Участвуя в НТО, школьники получают возможность работать с практикоориентированными задачами в области прорывных технологий, собирать команды единомышленников, включаться в профессиональное экспертное сообщество, а также заработать льготы для поступления в вузы.

У НТО есть площадки подготовки по всей стране, которые занимаются привлечением участников и проводят мероприятия по подготовке к соревнованиям. Они могут быть открыты:

- в организациях общего и дополнительного образования;
- на базе частных кружков в области программирования, робототехники и иных технологий;
- в вузах;
- технопарках

и других организациях.

Каждое образовательное учреждение, ученики которого участвуют в НТО или НТО Junior, может стать площадкой подготовки к олимпиаде, что дает возможность включиться в Кружковое движение НТИ.

На сайте НТО размещены инструкции о том, как организация может стать площадкой подготовки: <https://ntcontest.ru/mentors/stat-ploshadkoi/>. Условия регистрации и требования к работе площадок подготовки обновляются вместе с развитием олимпиады. Обновленная версия размещается на сайте перед началом нового цикла олимпиады.



Наставники НТО

В НТО большое внимание уделяется работе с наставниками. Наставник НТО оказывает всестороннюю поддержку участникам Олимпиады, помогая решать организационные вопросы и развивать как технические знания и компетенции, так и социальные навыки, связанные с работой в команде.

Наставником может стать любой человек, которому интересно сопровождать участников и помогать им формировать необходимые для решения технологических задач компетенции и готовиться к соревнованиям. Это может быть преподаватель школы или вуза, педагог дополнительного образования, руководитель кружка, эксперт в технологической области, представитель бизнеса и т. п. Если наставнику не хватает собственных знаний, он может привлекать коллег и внешних экспертов и

поддерживать усилия и мотивацию учеников, которые разбирают задачи самостоятельно. На данный момент сообщество наставников НТО включает в себя более 7 тысяч человек.

Главная задача наставника — выстроить комплексную структуру подготовки к Олимпиаде в течение всего учебного года. В области ответственности наставника находится поддержка мотивации участников и помощь в решении возникающих проблем. Не менее важно зафиксировать цели и ожидания от предстоящих соревнований, что поможет оценить прирост профессиональных компетенций, личных и командных навыков за время подготовки.

Примеры организационных задач, которые стоят перед наставником НТО:

- Информирование и работа с мотивацией. На этапе регистрации на Олимпиаду наставник привлекает участников, рассказывая, что такое НТО и какие преимущества она предлагает. Наставнику необходимо разобраться в устройстве НТО, этапах и расписании этапов, а также изучить профили, чтобы помочь каждому ученику выбрать наиболее перспективные и интересные для него направления.
- Формирование программы подготовки. Наставник составляет график подготовки к НТО и следит за его реализацией, руководя процессом подготовки учеников.
- Отслеживание сроков. Наставник следит за сроками проведения этапов НТО и напоминает участникам о необходимости своевременной загрузки решений на платформу.

Примеры задач наставника, связанных с непосредственной подготовкой к соревнованиям:

- Анализ компетенций участников. Наставник вместе с учениками оценивает компетенции, которые необходимы для успешного участия в НТО, выявляет нехватку знаний и навыков и отбирает материалы и задачи, которые ученикам нужно изучить и решить.
- Содержательная подготовка к первому и второму отборочному этапу. Наставник вместе с учениками изучает материалы для подготовки, рекомендованные разработчиками выбранных профилей, а также разбирает и решает задачи НТО прошлых сезонов. Рекомендуется использовать записи вебинаров, материалы и онлайн-курсы профилей.
- Содержательная подготовка к заключительному этапу. Наставник может использовать разборы задач заключительного этапа прошлых лет, а также следить за расписанием подготовительных очных и дистанционных мероприятий и рекомендовать ученикам их посещать.

Примеры задач наставника в области развития социальных навыков, связанных с развитием личной эффективности и взаимодействия с другими участниками:

- Формирование команд. Второй отборочный этап НТО проходит в командном формате. Наставник помогает ученикам сформировать эффективную команду с оптимальным распределением ролей. В ряде случаев он может содействовать в поиске недостающих участников команды, в том числе в других городах и стать наставником такой команды, коммуникация в которой осуществляется через web-сервисы.
- Отслеживание прогресса и анализ полученного опыта. Наставник проводит ре-

флексию прогресса отдельных участников и команды по результатам каждого этапа НТО и после завершения участия в соревнованиях. Это помогает участникам оценить свое движение по траектории соревнований, сильные и слабые стороны, сформулировать, каких компетенций не хватило для более высокого результата и как их можно улучшить в будущем.

- Поддержка и мотивирование участников. Наставник поддерживает интерес учеников к соревнованиям, а также помогает им сохранять высокую мотивацию, что особенно важно, если команда показала результаты хуже, чем ожидалось.
- Выстраивание индивидуальной образовательной траектории. Наставник может помочь ученикам осознанно создать собственную траекторию развития, в том числе вне НТО: подбор обучающих курсов и соревнований, выбор вуза и направления дальнейшего обучения.

Поддержка наставников НТО

Работе наставников посвящен отдельный раздел на сайте НТО: <https://ntcontest.ru/mentors/>.



Для систематизации знаний и подходов к работе наставников в рамках инженерных соревнований разработан курс «Дао начинающего наставника: как сопровождать инженерные команды»: <https://stepik.org/course/124633/promo>. Курс формирует общие представления о работе наставников в области подготовки участников к инженерным соревнованиям.



Для совершенствования профессиональных компетенций по направлениям профилей разработан курс «Дао наставника: как развивать технологические компетенции»: <https://stepik.org/course/186928/promo>.



Наставникам для ведения занятий с учениками предлагаются образовательные программы, разработанные на основе восьмилетнего опыта организации подготовки к НТО. В настоящий момент такие программы представлены по 10-ти передовым технологическим направлениям:

- компьютерное зрение;
- геномное редактирование;
- водная, летающая и интеллектуальная робототехника;
- машинное обучение и искусственный интеллект;
- нейротехнологии;
- беспроводная связь, дополненная реальность;

и др.

<https://ntcontest.ru/mentors/education-programs/>.



Регистрируясь на платформе НТО, наставники получают доступ к личному кабинету, в котором отображается расписание отборочных соревнований и мероприятий по подготовке, требования к знаниям и компетенциям при решении задач отборочных этапов.

Формируется сообщество наставников НТО. Ежегодно Кружковое движение НТИ проводит Всероссийский конкурс технологических кружков: <https://konkurs.kruzhok.org>, принять участие в котором может каждый наставник. По итогам конкурса кружки-участники размещаются на Всероссийской карте кружков: <https://map.kruzhok.org>.



В 2022 году был разработан Навигатор для наставников команд или отдельных участников НТО: <https://www.notion.so/bd1v/5a1866975c2744728c2bd8ba80d21ec2>.



Навигатор ориентирован на начинающих наставников и помогает погрузиться в работу с НТО. Опытным наставникам Навигатор может быть полезен как сборник важных рекомендаций и статей:

- Смогут ли мои ученики принять участие в НТО.
- Как наставнику зарегистрироваться в НТО.
- Как помочь участникам выбирать профили.
- Что можно успеть сделать, если я и мои ученики начнем участвовать с нового учебного года.
- Как убедить руководство включиться в НТО.
- Что важно знать, начиная подготовку школьников.
- Как организовать подготовку.
- Как проводить рефлексию.
- Как мотивировать участников.
- Как работать с командой участников НТО.

Организаторы Олимпиады также оказывают экспертно-методическую поддержку сообществу наставников. Были разработаны методические рекомендации для наставников: «Технологическая подготовка инженерных команд»: <https://journal.kruzhok.org/tpost/pggs3bp7y1-tehnologicheskaya-podgotovka-inzhenernih>. Рассмотрены особенности подготовки к 5-ти направлениям:

- Большие данные.
- Машинное обучение.

- Искусственный интеллект.
- Спутниковые системы.
- Летящая робототехника.



Для наставников НТО разработан и постоянно пополняется страница с материалами для профессионального развития: <http://clc.to/for-mentor>.



Беспилотные авиационные системы

Профиль «Беспилотные авиационные системы» нацелен на вовлечение школьников 8–11 классов в техническую и инновационную деятельность в области проектирования систем автоматического управления беспилотными летательными аппаратами (БЛА). Особое внимание уделяется организации процесса разработки автопилота, при котором возможна практическая проверка разработанных программ на симуляторе полета БЛА самолетного типа «UAVIANT».

Профиль направлен на решение следующих технологических барьеров Национальной технологической инициативы:

- сенсоры и преобразующая аппаратура оптического, теплового, гиперспектрального, радиолокационного зондирования поверхности, радиолокационные станции бортового обзора, в том числе с функцией распознавания образов людей, животных, транспортных средств и потоков, мобильных и стационарных объектов для обеспечения мониторинга, подсчета наблюдаемых объектов и выявления их характерных признаков, а также для выявления признаков чрезвычайных ситуаций;
- беспроводная система мониторинга систем и узлов БВС;
- электронно-оптическая система «улучшенного видения» для улучшения изображения в условиях тумана, плохой видимости, при наличии препятствий;
- беспилотное воздушное судно для сбора, хранения и обработки информации о характеристиках окружающего пространства.

Основной задачей заключительного этапа данного профиля является разработка системы автоматического управления БЛА самолетного типа для доставки медицинского оборудования и медикаментов в автоматическом режиме за ограниченное время до адресатов, находящихся в удаленных или труднодоступных местах. Участники не только разрабатывают систему автоматического управления для БЛА, но и автоматизируют процесс обработки полученных данных об исследуемом объекте.

Акцент при разработке командной комплексной инженерной задачи направлен на четкое разделение основной задачи на подзадачи, при успешном решении которых участники достигали поставленной цели. Каждая подзадача на заключительном этапе имеет свой уровень сложности и оценивается соответствующим баллом. Подзадачи распределены по соответствующим этапам.

Первый отборочный этап включает в себя решение олимпиадных задач предметного и инженерного туров. Предметный тур проводится по двум школьным предметам: информатика и физика. Хорошие знания по данным предметам крайне необходимы участникам профиля, так как требуются для решения задач второго отборочного и заключительного этапов. В рамках инженерного тура, основной задачей которого является погружение участников в тематику профиля, участники также решают задачи по математике, физике и информатике, а также проходят тестирование по основным понятиям, связанным с беспилотными авиационными системами.

На втором отборочном этапе участники решают задачи по определению траектории полета, расчету аэродинамики груза, разработке одного из каналов системы управления БЛА, а также по обработке изображений. Задачи носят междисциплинарный характер.

нарный характер и в упрощенной форме воссоздают элементы решения комплексной инженерной задачи заключительного этапа.

На заключительном этапе все полученные в предыдущих этапах знания участники используют при работе с реальным БЛА самолетного типа (работа с датчиками и органами управления БЛА с использованием специального стенда полунатурного моделирования, работа с системой технического зрения, моделирование полета и проверка своих решений при помощи специально разработанного симулятора). Такой подход дает возможность участникам глубоко понять объект исследования и принцип его работы.

Участники профиля «Беспилотные авиационные системы» поступают в ведущие вузы России, в том числе в МАИ, на специальности, связанные с авиационными технологиями, управлением в технических системах и другими. Участие школьников в Олимпиаде по профилю «Беспилотные авиационные системы» позволяет не только повысить популярность авиационной отрасли среди школьников, но и обеспечить профильные инженерные вузы талантливыми студентами, формируя таким образом кадровый потенциал нашей страны в одном из важных для ее развития технологических направлений.

Работа наставника НТО на первом отборочном этапе

На первом отборочном этапе НТО участникам предлагаются задачи по предметам, соответствующим выбранным профилям. Для подготовки к первому отборочному этапу Олимпиады наставник может использовать следующие рекомендуемые форматы и мероприятия:

- Разбор задач первого отборочного этапа НТО прошлых лет.
- Мини-соревнования по решению задач предметных олимпиад муниципального уровня.
- Углубленные занятия по разделам предметов в соответствии с рекомендациями разработчиков профилей.

Для проверки, самостоятельного решения или проведения мини-соревнований могут использоваться предметные курсы НТО на платформе Stepik. Также возможно привлечение других преподавателей-предметников для проведения занятий в случае, если у наставника недостаточно компетенций в области предметных олимпиад.

Инженерный тур состоит из курса или теоретических материалов, погружающих участников в тематику профиля, и теоретических и практических заданий, как правило связанных с теорией.

Первый отборочный этап

Предметный тур. Информатика и программирование

Первая волна. Задачи 8–11 класса

Задача II.1.1.1. Поздравление в конверте (10 баллов)

Темы: задачи для начинающих.

Условие

Алиса хочет поздравить Боба с днем рождения. Она взяла прямоугольный лист бумаги размера $a \times b$ и написала на нем поздравление в стихах. У Алисы есть красивый конверт тоже прямоугольной формы размером u на v . Алиса хочет положить свое поздравление в этот конверт. Однако лист может не войти в конверт. В этом случае Алиса готова сложить лист пополам вдоль одной из сторон, чтобы поместить его в конверт. Обратите внимание, что Алиса может сделать не более одного сгиба. Лист можно поворачивать, но одна из сторон листа должна быть параллельной одной из сторон конверта.

Напишите программу, которая определит, сможет ли Алиса уложить лист в конверт по указанным правилам.

Мы будем считать, что лист входит в конверт, если сторона листа будет строго меньше соответствующей стороны конверта.

Формат входных данных

На вход в первой строке подается два натуральных числа a и b — длины сторон листа. Во второй строке на вход подаются натуральные числа u и v — размеры конверта. Все числа не превосходят 1000.

В языке Python прочитать два целых числа, записанных в одной строке можно, используя следующий код.

```
a, b = map(int, input().split())
```

Формат выходных данных

Если поздравление можно вложить в конверт без сгиба, то следует вывести число 0. Иначе, если поздравление можно вложить в конверт сделав один сгиб, то следует вывести число 1. В остальных случаях следует вывести число -1 .

Методика проверки

Программа проверяется на 20-ти тестах. Прохождение каждого теста оценивается в 0,5 балла. Тесты из условия задачи при проверке не используются.

Примеры

Пример №1

Стандартный ввод
120 200 130 250
Стандартный вывод
0

Пример №2

Стандартный ввод
120 200 110 130
Стандартный вывод
1

Пример №3

Стандартный ввод
400 100 200 150
Стандартный вывод
-1

Решение

В этой задаче требуется аккуратно написать требуемые по условию логические выражения. Их запись существенно упростится, если упорядочить длины так, чтобы всегда имел место инвариант $a \leq b$ и $u \leq v$.

Тогда для проверки возможности вложения листа в конверт меньшую сторону листа следует всегда сравнивать с меньшей стороной конверта.

Для проверки возможности вложения листа в конверт после сгиба надо поочередно поделить меньшую и большую сторону на два и использовать такую же проверку. Следует не забыть, что после деления длины большей стороны на два, она может стать меньше, чем меньшая сторона.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```

1 a, b = sorted(list(map(int, input().split())))
2 u, v = sorted(list(map(int, input().split())))
3 if a<u and b<v:
4     print(0)
5 elif a/2<u and b<v or a<u and b/2<v or b/2<u and a<v:
6     print(1)
7 else:
8     print(-1)

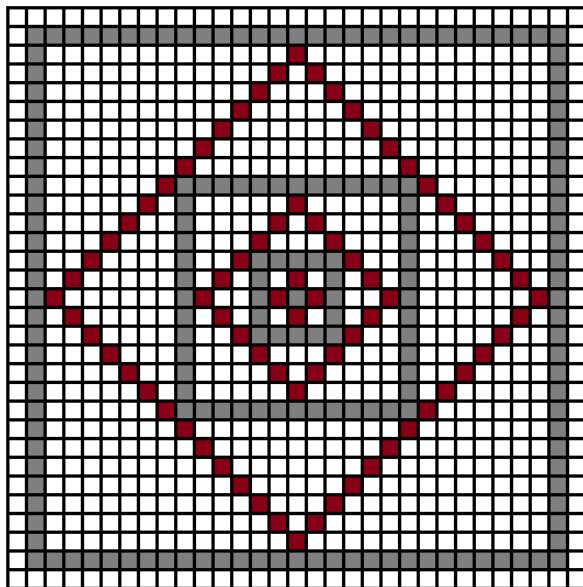
```

Задача II.1.1.2. Квадраты (15 баллов)

Темы: задачи для начинающих, комбинаторика.

Условие

У Алисы и Боба есть прямоугольный лист бумаги в клеточку. Они по очереди рисуют квадраты, закрашивая некоторые из клеточек. Алиса рисует квадраты, ориентируя их вдоль сторон листа, а Боб — под углом в 45° . При этом Алиса рисует первый квадрат из одной клеточки, а каждый новый квадрат описывается вокруг предыдущего. Для лучшего понимания смотрите рисунок. Серым цветом на нем нарисовано четыре квадрата Алисы, а коричневым нарисовано три квадрата Боба. Всего семь квадратов.



Алиса и Боб вместе нарисовали n квадратов. Напишите программу, которая определит, сколько клеточек на листе бумаги будет закрашено.

Формат входных данных

На вход подается единственное натуральное число n — количество квадратов, $1 \leq n \leq 100$.

Формат выходных данных

Выведите одно натуральное число — суммарное количество закрашенных клеточек.

Методика проверки

Программа проверяется на 15-ти тестах. Прохождение каждого теста оценивается в 1 балл. Тесты из условия задачи при проверке не используются.

Примеры*Пример №1*

Стандартный ввод
7
Стандартный вывод
253

Пример №2

Стандартный ввод
2
Стандартный вывод
5

Решение

Заметим, что по условию задачи количество квадратов не превышает 100, поэтому посчитаем, из скольких клеточек состоит каждый квадрат, и просуммируем полученные значения в цикле.

Обратим внимание на количество клеточек на стороне квадрата. Легко заметить и доказать, что если предыдущий квадрат был серый, и его сторона содержала k клеточек, то сторона следующего за ним коричневого квадрата будет содержать $k + 1$ клеточку. Если же предыдущий квадрат был коричневым, и его сторона содержала k клеточек, то сторона следующего серого квадрата будет содержать $2k + 1$ клеточку.

В приведенной ниже программе в переменной `ln` хранится количество клеточек, из которых состоит одна сторона текущего квадрата. В переменной `ans` накапливается сумма.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```

1  n = int(input())
2  ans = 1
3  ln = 1
4  for i in range(n):
5      ans += (ln - 1) * 4
6      if i%2==0:
7          ln += 1
8      else:
9          ln = 2 * ln + 1
10 print(ans)

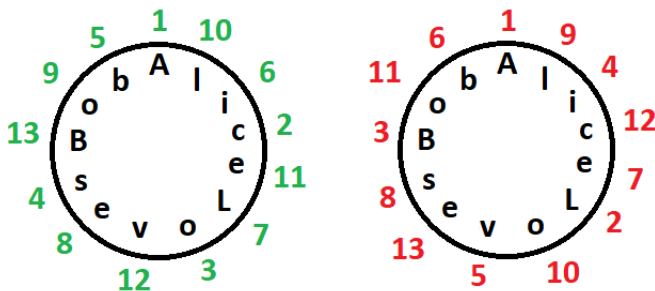
```

Задача II.1.1.3. Две строки (15 баллов)

Темы: строки, структуры данных.

Условие

У Алисы и Боба есть секретная информация, которая записана в виде строки s . Чтобы сохранить секрет Алиса сделала перестановку символов в строке по следующему правилу. Она записала все символы строки по кругу, потом записала в ответ первый символ и далее стала выписывать символы из кольца через два.



Рассмотрим пример. Пусть секретная строка — *AliceLovesBob*. Алиса запишет эту строку, как показано на рисунке. Далее она выпишет первую букву строки A , пропустит два следующих символа, напишет букву s , пропустит еще два символа, напишет букву o и так далее по кругу. В результате у нее будет записана строка *AcosbiLeolevB*. Номера на рисунке слева соответствуют последовательности перечисления букв Алисой.

Боб шифрует эту же строку таким же алгоритмом, однако, в отличие от Алисы, он пропускает по четыре буквы, а не по две. Номера на рисунке справа соответствуют последовательности перечисления букв Бобом. Таким образом Боб получит строку *ALBivbeslooce*.

Боб был небрежен и потерял зашифрованную строку, однако у него есть строка, зашифрованная Алисой. Напишите программу, которая по зашифрованной строке Алисы найдет зашифрованную строку Боба.

Формат входных данных

На вход подается одна непустая строка — шифр Алисы. Строка состоит только из строчных и заглавных символов латиницы. Длина строки не превосходит 1000 и не кратна трем и пяти.

Формат выходных данных

Выведите одну строку — шифр Боба.

Методика проверки

Программа проверяется на 15-ти тестах. Прохождение каждого теста оценивается в 1 балл. Тест из условия задачи при проверке не используется.

Примеры

Пример №1

Стандартный ввод
AcosbiLeolevB
Стандартный вывод
ALBivbeslooce

Решение

Решение задачи состоит из двух частей. В первой части требуется восстановить исходную строку по коду Алисы. Для этого можно сделать список из символов нужной длины и каждый i -тый символ из кодовой строки записывать в позицию $3i \bmod n$, где n — длина строки. Операция взятия остатка от деления здесь используется для движения по кольцу.

Во второй части при помощи аналогичного приема полученная строка кодируется по обратной формуле с множителем 5.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```

1 s = input()
2 n = len(s)
3 tmp = [''] * n
4 ans = ''
5 for i in range(n):
6     tmp[(i * 3) % n] = s[i]
7 for i in range(n):
8     ans += tmp[(i * 5) % n]
9 print(ans)

```

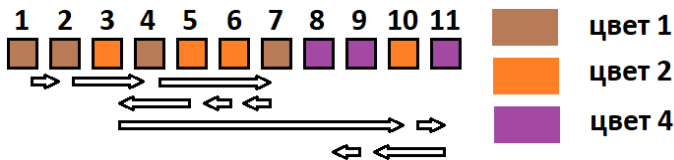
Задача П.1.1.4. Покраска кубиков (30 баллов)

Темы: реализация, сортировки, структуры данных, динамическое программирование, комбинаторика.

Условие

На ленте в один ряд расставлено n кубиков. Каждый кубик необходимо покрасить в определенный цвет. Все цвета пронумерованы числами от 1 до k . Покраска выполняется роботом, который может перемещаться от одного кубика к другому и красить один выбранный кубик в определенный цвет. Конструктивно робот устроен так, что он может сначала красить кубики в цвет номер 1, затем в цвет номер 2 и так далее в порядке возрастания. Вернуться к цвету с меньшим номером после того, как был выбран цвет с большим номером, нельзя.

Среди всего прочего робот тратит время на перемещение между кубиками. Будем считать, что перемещение между двумя соседними кубиками занимает ровно одну с. Требуется составить последовательность действий для робота, в которой время, затраченное на перемещение между кубиками, будет минимально возможным.



Рассмотрим пример на схеме. Имеется 11 кубиков, которые надо покрасить в три цвета с номерами 1, 2, 4. Робот начнет движение от кубика номер 1 направо к кубику номер 2 (1 с), далее к кубику с номером 4 (2 с), и наконец к кубику номер 7 (3 с). После этого робот меняет цвет. Будет выгоднее, если робот начнет сначала двигаться влево. Он пройдет к кубику номер 6 (1 с), далее к кубику номер 5 (1 с), далее к кубику номер 3 (2 с). После этого он развернется и пойдет к кубику номер 10 (7 с). Далее робот сменит цвет на 4, так как нет кубиков, которые требуется красить в цвет 3, и пойдет направо к кубику номер 11 (1 с). После этого он развернется и пойдет к кубику номер 9 (2 с) и наконец к кубику номер 8 (1 с). В этот момент робот остановит работу, затратив на перемещения суммарно 21 с.

Обратите внимание, что по условию задачи робот может выбрать цвет 4 только после цвета 2. Существуют другие возможные маршруты движения робота, но они займут больше времени.

Напишите программу, которая найдет минимально возможное суммарное время перемещения робота между кубиками до момента пока все они не будут покрашены. Изначально робот находится у кубика номер 1.

Формат входных данных

На вход в первой строке подается два натуральных числа n и k — количество кубиков и количество цветов, $1 \leq n, k \leq 100000$. Во второй строке на вход подается n натуральных чисел c_1, c_2, \dots, c_n , где c_i — требуемый цвет i -го кубика, $1 \leq c_i \leq k$.

Формат выходных данных

Выведите одно число — минимально возможное время перемещения между кубиками.

Методика проверки

Программа проверяется на 60-ти тестах. Прохождение каждого теста оценивается в 0,5 балла. Тест из условия задачи при проверке не используется. Ниже в таблице приведены возможные тестовые случаи.

Тестовый случай	Номера тестов
$n = k; n \leq 1000$; все c_i различны.	1–8
$n = k; n \leq 100000$; все c_i различны.	9–20
$k = 3; n \leq 1000$	21–24
$k = 4; n \leq 1000$	25–30
$k = 5; n \leq 1000$	31–40
$n \leq 1000$	41–50
$n \leq 100000$	51–60

Примеры

Пример №1

Стандартный ввод
11 4
1 1 2 1 2 2 1 4 4 2 4
Стандартный вывод
21

Решение

Данная задача может быть решена методом динамического программирования. Пусть, начиная покраску кубиков в цвет i , робот находится в некоторой точке x_i , причем самый левый из кубиков этого цвета находится в точке l_i , а самый правый — в r_i . Тогда оптимальным будет один из двух вариантов: из точки x_i пойти в r_i , а потом в l_i или сначала пойти в l_i , а потом в r_i . Таким образом, робот закончит покраску кубиков определенного цвета либо в точке l_i , либо в r_i , причем в первом случае время перемещения робота увеличится на $|x_i - r_i| + r_i - l_i$ с, а во втором — на $|x_i - l_i| + r_i - l_i$ с.

Обозначим за $f_l(i)$ и $f_r(i)$ — оптимальное время покраски кубиков в первые i цветов при условии, что робот остановится в точке l_i и r_i соответственно. Тогда можно определить следующие формулы:

$$\begin{aligned}
 f_l(0) &= 0; \\
 f_r(0) &= 0; \\
 f_l(i) &= r_i - l_i + \min(f_l(i-1) + |l_{i-1} - r_i|, f_r(i-1) + |r_{i-1} - r_i|); \\
 f_r(i) &= r_i - l_i + \min(f_l(i-1) + |l_{i-1} - l_i|, f_r(i-1) + |r_{i-1} - l_i|).
 \end{aligned}$$

В данных формулах находится время для двух вариантов перемещения: от самого левого и от самого правого кубика предыдущего цвета, после чего из полученных значений выбирается минимальное.

Программа будет содержать два цикла. В первом цикле для каждого цвета определяется местоположение самого левого и самого правого кубика, а во втором — выполняются вычисления по формулам.

При реализации программы надо учесть, что некоторые цвета могут отсутствовать. Поэтому в переменные `l` и `r` записываются координаты самого левого и правого кубика для предыдущего цвета.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```

1 n, k = map(int, input().split())
2 left = [n + 1] * (k + 1)
3 right = [0] * (k + 1)
4 i = 1
5 for col in map(int, input().split()):
6     left[col] = min(left[col], i)
7     right[col] = max(right[col], i)
8     i += 1
9 l, r, fl, fr = 1, 1, 0, 0
10 for i in range(1, k + 1):
11     if right[i] > 0:
12         dist = right[i] - left[i]
13         tl = dist + min(fl + abs(l - right[i]),
14                       fr + abs(r - right[i]))
15         tr = dist + min(fl + abs(l - left[i]),
16                       fr + abs(r - left[i]))
17         l, r, fl, fr = left[i], right[i], tl, tr
18 print(min(fl, fr))

```

Задача II.1.1.5. Обработка запросов (30 баллов)

Темы: реализация, структуры данных, два указателя, двоичный поиск.

Условие

Алиса проектирует вычислительную систему, предназначенную для обработки большого числа однотипных запросов. Проектируемая система будет содержать некоторое количество одинаковых процессоров. В каждый момент времени каждый процессор может быть либо свободен, либо занят обработкой ровно одного запроса. Продолжительность обработки является одинаковой для всех запросов и составляет s мс. Система должна работать в режиме реального времени, то есть каждый поступивший запрос должен незамедлительно передаваться на обработку любому свободному процессору.

Алиса хочет понять, сколько процессоров должна содержать вычислительная система. Для этого она собрала статистические данные о работе подобных систем в прошлом. Каждый набор данных содержит n чисел t_1, t_2, \dots, t_n , где t_i — момент

поступления запроса с номером i . Числа в наборе могут повторяться, однако они упорядочены по неубыванию. Если некоторый процессор приступил к обработке некоторого запроса в момент t_i , то в момент $t_i + s$ он сможет начать обрабатывать новый запрос.

Набор может содержать достаточно большой объем данных, поэтому от вас требуется написать программу, которая определит, какое минимальное число процессоров должно быть в вычислительной системе, чтобы все запросы были обработаны в момент их поступления.

Формат входных данных

На вход в первой строке подается два натуральных числа n и s — количество запросов и время обработки одного запроса, $1 \leq n \leq 200000$, $1 \leq s \leq 10^9$. Во второй строке записаны целые неотрицательные числа t_1, t_2, \dots, t_n , задающие моменты времени поступления запросов, $0 \leq t_1 \leq t_2 \leq \dots \leq t_n \leq 10^9$.

Формат выходных данных

Вывести одно число — минимальное количество процессоров, которое позволит обработать все запросы в момент их поступления.

Методика проверки

Программа проверяется на 30-ти тестах. Прохождение каждого теста оценивается в 1 балл. Тесты из условия задачи при проверке не используются. Ниже в таблице приведены возможные тестовые случаи.

Тестовый случай	Номера тестов
$n \leq 1000$; для всех t_i выполняется одно из двух условий: либо $t_i = t_{i+1}$, либо $t_i + s \leq t_{i+1}$.	1–5
$n \leq 1000$	6–15
$n \leq 200000$	16–30

Примеры

Пример №1

Стандартный ввод
9 30
90 90 90 120 120 120 120 200 200
Стандартный вывод
4

Пример №2

Стандартный ввод
10 30
0 25 110 125 125 130 140 140 140 155
Стандартный вывод
6

Пояснения к примерам

Первый пример соответствует первому тестовому случаю. В момент времени 120 приходит сразу четыре запроса, для обработки которых потребуется четыре процессора.

Расписание выполнения запросов во втором примере можно представить в следующей таблице.

Время поступления запроса	Время завершения обработки запроса	Номер процессора
0	30	1
25	55	2
110	140	1
125	155	2
125	155	3
130	160	4
140	170	1
140	170	5
140	170	6
155	175	2

Пять процессоров для своевременной обработки всех запросов будет уже недостаточно.

В задаче требуется найти такое число k , что для всех $i \leq n - k$ выполняется неравенство $t_{i+k} - t_i \leq s$. Действительно, пусть это утверждение имеет место. Тогда процессор, выполнявший задание с номером i , всегда сможет выполнить задание с номером $i+k$, и все задания можно выполнить своевременно, выдавая их процессорам по кругу. С другой стороны, пусть это утверждение не выполняется, то есть найдется такой номер j , что $t_{j+k} - t_j < s$. Тогда в момент времени t_{j+k} все k процессоров будут заняты исполнением запросов с номерами от j до $j + k - 1$, и все запросы не смогут быть выполнены своевременно.

Найти число k , для которого выполняется указанное утверждение можно при помощи двоичного поиска или метода двух указателей. Вариант решения с использованием двух вложенных циклов наберет лишь часть баллов, так как будет превышать ограничение по времени работы.

Метод двух указателей основан на использовании двух переменных `left` и `right`, которые используются в качестве индексов в массиве. Цикл строится таким образом, чтобы для каждого значения `left` находить минимальное значение `right` при котором `t[right] - t[left] >= s`.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```

1 n, s = map(int, input().split())
2 t = list(map(int, input().split()))
3 ans = 1
4 left = 0
5 right = 0
6 while right < n:
7     if t[right] - t[left] >= s:
8         left += 1
9     else:
10        right += 1
11    ans = max(ans, right - left)
12 print(ans)

```

Вторая волна. Задачи 8–11 класса

Задача II.1.2.1. Три мешка конфет (10 баллов)

Темы: задачи для начинающих, реализация.

Условие

Алиса и Боб получили в подарок три мешка конфет и они хотят поделить их поровну. Для этого Алиса возьмет некоторое количество конфет из каждого мешка, а остальные конфеты отдаст Бобу. Возможно, что из некоторого мешка Алиса возьмет все конфеты или не возьмет ни одной. Известно, что суммарное количество конфет является четным числом.

Напишите программу, которая определит, сколько конфет Алиса должна взять из каждого мешка, чтобы у нее оказалось ровно половина всех конфет. Программа может вывести любой правильный ответ.

Формат входных данных

На вход в первой строке подается три натуральных числа a , b и c — количество конфет в каждой кучке, $1 \leq a, b, c \leq 1000$.

В языке Python прочитать три целых числа, записанных в одной строке можно, используя следующий код.

```
a, b, c = map(int, input().split())
```

Формат выходных данных

Выведите в одной строке через пробел три целых неотрицательных числа — количество конфет, которое возьмет Алиса из каждого мешка.

Методика проверки

Программа проверяется на 20 тестах. Прохождение каждого теста оценивается в 0,5 балла. Тест из условия задачи при проверке не используется.

Примеры

Пример №1

Стандартный ввод
10 5 5
Стандартный вывод
7 3 3

Пояснения к примерам

Ответ 7 3 0 удовлетворяет всем требованиям. Но существует и множество других вариантов, например, 7 2 1 или 0 5 5.

Решение

Существует много способов составить требуемый набор чисел. Например, можно заметить, что если сумма трех чисел четная, то хотя бы одно из слагаемых тоже обязательно четное. Тогда это слагаемое можно поделить на два, еще одно поделить на два с округлением вниз, а последнее — с округлением вверх.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```

1 a, b, c = map(int, input().split())
2 if a % 2 == 0:
3     print(a // 2, b // 2, (c + 1) // 2)
4 else:
5     print((a + 1) // 2, b // 2, c // 2)

```

Задача II.1.2.2. Трехцветная сортировка (15 баллов)

Темы: задачи для начинающих, структуры данных.

Условие

У Алисы есть упорядоченный набор карточек, каждая из которых раскрашена в один из трех цветов: красный, зеленый, синий. Кроме того, на каждой карточке записано некоторое натуральное число. Алиса хочет выполнить сортировку чисел, чтобы сначала шли все числа на красных карточках, далее — на зеленых и наконец — на синих. При этом взаимное расположение карточек одного цвета не должно измениться. Например, если в исходном наборе было две красных карточки с числами 20

и 10, причем карточка с числом 20 располагалась раньше, чем карточка с числом 10, то после упорядочивания 20 по-прежнему должна находиться раньше, чем 10.

Напишите программу, которая отсортирует карточки в требуемом порядке.

Формат входных данных

На вход в первой строке подается последовательность символов c_1, c_2, \dots, c_n , где c_i задает цвет i -той карточки и может принимать одно из трех значений r , g или b . Каждый из символов обозначает определенный цвет: r — красный, g — зеленый, b — синий. Символы записаны без пробелов и других разделителей, $1 \leq n \leq 1000$.

Во второй строке записана последовательность натуральных чисел a_1, a_2, \dots, a_n , где a_i задает число, записанное на i -той карточке. Все числа различны и не превосходят n .

Формат выходных данных

В одной строке через пробел вывести требуемую последовательность чисел после сортировки.

В языке Python для вывода чисел в цикле на одной строке через пробел можно использовать следующую команду.

```
print(x, end=' ')
```

Методика проверки

Программа проверяется на 15-ти тестах. Прохождение каждого теста оценивается в 1 балл. Тесты из условия задачи при проверке не используются.

Примеры

Пример №1

Стандартный ввод
bbb 3 1 2
Стандартный вывод
3 1 2

Пример №2

Стандартный ввод
rgrg 4 1 2 3
Стандартный вывод
4 2 1 3

Пример №3

Стандартный ввод
brrg 1 2 3 4
Стандартный вывод
2 3 4 1

Пояснения к примерам

В первом примере все карточки одного цвета, поэтому упорядочивать нечего.

Во втором примере карточки 4 и 2 красного цвета, поэтому они окажутся в начале, сохранив взаимное расположение. Карточки 1 и 3 зеленого цвета, поэтому они сдвинутся в конец, также сохранив взаимное расположение.

В третьем примере в начале последовательности будут красные карточки 2 и 3, далее зеленая 4, далее синяя 1.

Решение

Для решения этой задачи достаточно сохранить цвета и номера карточек в списке. Далее в трех циклах вывести сначала номера красных карточек, потом — зеленых, и наконец, — синих.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```

1 s = input()
2 p = list(map(int, input().split()))
3 for a in 'rgb':
4     for i in range(len(s)):
5         if s[i] == a:
6             print(p[i], end = ' ')

```

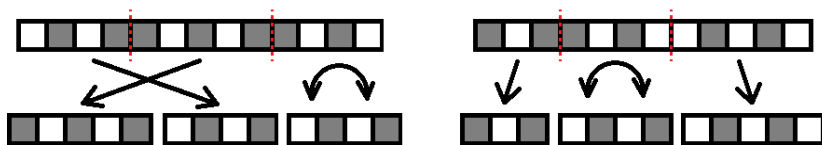
Задача II.1.2.3. Черно-белая полоска (15 баллов)

Темы: задачи для начинающих, реализация, строки.

Условие

У Алисы есть полоска бумаги, расчерченная на клеточки. Полоска имеет ширину в одну клеточку и длину в n клеточек. Алиса хотела раскрасить каждую клеточку в белый или черный цвет так, чтобы клеточки разных цветов чередовались. Но после того, как вся полоска была раскрашена, выяснилось, что Алиса ошиблась, и существует ровно две непересекающихся пары соседних клеточек, раскрашенных в один цвет. Отметим, что *три и более клеток подряд не могут иметь один цвет*. Чтобы исправить свои ошибки, Алиса решила разрезать полоску в двух местах, переставить

и, возможно, развернуть полученные три части, а затем склеить их. На рисунке ниже показаны два примера разрезания и склейки полоски.



В примере на картинке слева исходная полоска разрезается на три части и склеивается в следующем порядке. Кусочек из середины от с номерами клеток из диапазона [5; 9] становится самым левым. Далее к нему пристыковывается кусочек с номерами клеток [1; 4]. И, наконец, справа пристыковывается кусочек с номерами клеток [10; 13], который при этом разворачивается на 180° . В результате будет получена полоска из клеток с чередующимися цветами.

В примере на картинке справа кусочки полоски остаются на своих местах, но средняя полоска с номерами клеток [4; 7] разворачивается на 180° .

Напишите программу, которая определит, сможет ли Алиса указанным способом сделать полоску из клеточек чередующихся цветов и, если это возможно, то составит схему разрезания существующей полоски на три кусочка и склейки этих кусочков. Полученная полоска может начинаться как с клетки белого, так и черного цвета. Если требуемую полоску можно получить различными способами, то в качестве ответа можно взять любой из них.

Формат входных данных

На вход подается одна строка, описывающая вид исходной полоски. Строка состоит из символов w и b , обозначающих клетку белого и черного цвета соответственно. Длина строки не превосходит 1000. Гарантируется, что строка имеет вид, описанный в условии задачи.

Формат выходных данных

Если составить полоску из клеток чередующихся цветов невозможно, то программа должна вывести единственное слово *no*. В противном случае вывод должен содержать ровно три строки, каждая из которых описывает кусочек исходной ленты в виде трех чисел. Первое и второе число задают номера начальной и конечной клетки кусочка соответственно. Третье число может иметь одно из двух значений — 0 или 180 в зависимости от того, поворачивается кусочек на 180° или нет. Строки должны следовать в порядке склейки кусочков слева направо.

Методика проверки

Программа проверяется на 15-ти тестах. Прохождение каждого теста оценивается в 1 балл. Тесты из условия задачи при проверке не используются.

Примеры

Пример №1

Стандартный ввод
wbwbwbwbwbwbw
Стандартный вывод
5 9 0
1 4 0
10 13 180

Пример №2

Стандартный ввод
bwbwbwbwbwbw
Стандартный вывод
1 3 0
8 12 180
4 7 0

Пример №3

Стандартный ввод
bbwbw
Стандартный вывод
no

Пример программы-решения

Ниже представлено решение на языке Python 3.

```

1 s = input()
2 n = len(s)
3 x = []
4 for i in range(1, n):
5     if s[i] == s[i-1]:
6         x.append(i)
7 if s[x[0]] != s[x[1]]:
8     print(1, x[0], 0)
9     print(x[0] + 1, x[1], 180)
10    print(x[1] + 1, n, 0)
11 elif s[x[0]] == s[0] or s[x[0]] == s[-1]:
12    print('no')
13 else:
14    print(x[0] + 1, x[1], 0)
15    print(1, x[0], 0)
16    print(x[1] + 1, n, 180)

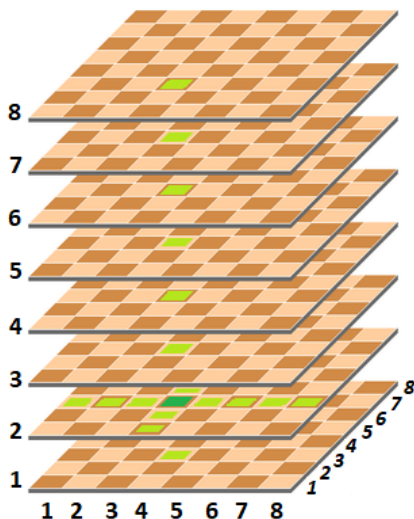
```

Задача II.1.2.4. Расстановка ладей на трехмерной шахматной доске (30 баллов)

Темы: реализация, сортировки, структуры данных, динамическое программирование, комбинаторика.

Условие

Алиса и Боб учатся пространственному воображению и решают для этого математические головоломки на трехмерной шахматной доске размера n . Такая доска состоит из n двумерных квадратных досок, расположенных друг над другом. На рисунке изображена трехмерная шахматная доска размера 8.



Каждую клетку трехмерной доски можно задать тремя целыми числами из диапазона $[1; n]$: порядковым номером двумерной доски, номером вертикали на двумерной доске и номером горизонтали. Например, клетка, выделенная на рисунке темно-зеленым цветом, задается тройкой чисел $(2, 4, 3)$.

Трехмерная шахматная ладья ходит по двумерной доске по стандартным правилам, то есть за один ход может переместиться на любую клетку в той же вертикали или горизонтали, где она находится. Вместе с тем трехмерная ладья может за один переход перейти на любую другую доску в клетку с такой же двумерной координатой. На рисунке светло-зеленым цветом показаны клетки в которые может перейти ладья из клетки с координатами $(2, 4, 3)$. В этом случае говорят, что ладья бьет эти клетки.

Алиса и Боб уверены, что на трехмерной шахматной доске размера n можно расставить n^2 ладей так, что они не будут бить друг друга, но никак не могут понять принцип расстановки.

Напишите программу, которая найдет любую допустимую расстановку ладей на трехмерной шахматной доске размера n так, чтобы они не били друг друга.

Формат входных данных

На вход подается единственное натуральное число n — размер доски, $1 \leq n \leq 30$.

Формат выходных данных

Выведите координаты n^2 клеток, на которых будут расположены ладьи. Три координаты каждой клетки выводятся через пробел в отдельной строке. Порядок перечисления клеток может быть произвольным.

Методика проверки

Программа проверяется на 30-ти тестах. Номер теста совпадает с числом n .

Примеры

Пример №1

Стандартный ввод
3
Стандартный вывод
1 2 2 1 3 3 2 1 3 2 2 1 2 3 2 3 1 2 3 2 3 3 3 1

Пояснения к примеру

Расстановка ладей из примера показана на рисунке ниже.



Пример программы-решения

Ниже представлено решение на языке Python 3.

```

1 n = int(input())
2 for i in range(n):
3     for j in range(n):
4         print(i + 1, j + 1, (i + j) % n + 1)

```

Задача II.1.2.5. Удаление скобок (30 баллов)

Темы: математика, строки, рекурсивные алгоритмы.

Условие

Боб любит формализм во всем, включая запись математических выражений, поэтому при их записи он ставит скобки так, чтобы каждая операция выделялась своей парой скобок. Например, выражение $(a + b + c) * d$ он запишет как $((a + b) + c) * d$ или как $((a + (b + c)) * d)$, а выражение $a * b + c * d$ как $((a * b) + (c * d))$. Таким образом, количество пар скобок в выражениях Боба всегда равно количеству операций, а для операции, которая выполняется последней, пара скобок всегда ограничивает все выражение.

Алиса считает такой перфекционизм избыточным и старается ставить скобки только там, где они нужны для правильных вычислений. Например, в выражении $(a + b + c) * d$ скобки убрать уже нельзя, поскольку выражение $a + b + c * d$ по математическим правилам задает другой порядок применения операций, и результаты вычисления этих двух выражений могут различаться. А вот в выражении $a + (b + c)$ скобки убрать уже можно, поскольку для сложения имеет место сочетательное свойство или, как говорят математики, аксиома ассоциативности. Также можно убрать скобки и в выражении $(a * b) + (c * d)$, поскольку по договоренностям умножение выполняется раньше, чем сложение.

Напишите программу, которая перепишет выражение, записанное Бобом, в тот вид, который нравится Алисе. Обратите внимание, что программа должна просто убрать все избыточные скобки. Другие преобразования делать нельзя. Полученное выражение должно иметь результат вычислений, совпадающий с результатом исходного выражения, при любых значениях параметров.

Формат входных данных

На вход в единственной строке поступает правильно записанное арифметическое выражение, состоящее из имен параметров, скобок и операций «+» и «*». Каждый параметр записывается в виде одной строчной буквы латиницы. Имена параметров не повторяются и встречаются в алфавитном порядке, таким образом, количество операций не превосходит 25. Выражение содержит как минимум одну операцию. Каждая операция в выражении выделяется своей парой скобок, как записано в условии задачи.

Формат выходных данных

Программа должна вывести исходное выражение без избыточных скобок. Порядок следования параметров в ответе должен совпадать с порядком в исходном выражении. В частности, это означает что для теста $(a + b)$ ответ $b + a$ будет считаться ошибочным.

Методика проверки

Программа проверяется на 30-ти тестах. Прохождение каждого теста оценивается в 1 балл. Выражения в первых девяти тестах содержат не более двух операций. Тесты из условия задачи при проверке не используются.

Примеры

Пример №1

Стандартный ввод
(a+b)
Стандартный вывод
a+b

Пример №2

Стандартный ввод
((a+(b+c))*d)
Стандартный вывод
(a+b+c)*d

Пример №3

Стандартный ввод
((a*b)+(c*d))
Стандартный вывод
a*b+c*d

Пример программы-решения

Ниже представлено решение на языке Python 3.

```

1 def parse(s,i):
2     if s[i] != '(':
3         return (s[i], i + 1, s[i])
4     else:
5         left, i, lop = parse(s, i + 1)
6         op = s[i]
7         right, i, rop = parse(s, i + 1)
8         if op == '*' and lop == '+':
9             left = '(' + left + ')'
10        if op == '*' and rop == '+':
11            right = '(' + right + ')'
12        return (left + op + right, i + 1, op)
13 print(parse(input(), 0)[0])

```

Третья волна. Задачи 8–11 класса

Задача II.1.3.1. Кодировка подмножеств (10 баллов)

Темы: задачи для начинающих, математика, реализация.

Условие

Недавно Алиса узнала об одном способе закодировать одним целым числом любое подмножество некоторого заданного конечного множества. Для этого необходимо сопоставить каждому элементу множества число, равное некоторой степени двойки. Теперь в качестве кода произвольного подмножества можно взять сумму чисел, соответствующих элементам этого подмножества.

Алиса составила множество из шести своих друзей и поставила им в соответствие последовательные степени двойки:

- 1 — *Anna*;
- 2 — *Boris*;
- 4 — *Cary*;
- 8 — *David*;
- 16 — *Eva*;
- 32 — *Fiona*.

Например, подмножество $\{Anna, Cary, Eva, Fiona\}$ будет закодировано числом 53. ($1 + 4 + 16 + 32 = 53$).

Алиса тренируется быстро декодировать подмножество по его коду. Напишите программу, которая позволит проверить ее навыки. Программа должна получать на вход некоторый код и выводить имена друзей, входящих в подмножество с этим кодом.

Формат входных данных

На вход подается единственное натуральное число n — код подмножества, $1 \leq n \leq 63$.

Формат выходных данных

Выведите имена друзей Алисы, которые входят в закодированное подмножество. Каждое имя следует выводить в отдельной строке. Порядок имен может быть произвольным.

Ниже приведен фрагмент программы на языке Python в котором создается список с правильным написанием слов.

```
Names = ['Anna', 'Boris', 'Cary', 'David', 'Eva', 'Fiona']
```

Методика проверки

Программа проверяется на 20-ти тестах. Прохождение каждого теста оценивается в 0,5 балла. Тест из условия задачи при проверке не используется. Первые шесть тестов — это последовательные степени двойки от 1 до 32.

Примеры

Пример №1

Стандартный ввод
53
Стандартный вывод
Anna Cary Eva Fiona

Пример программы-решения

Ниже представлено решение на языке Python 3.

```

1 Names = ['Anna', 'Boris', 'Cary', 'David', 'Eva', 'Fiona']
2 n = int(input())
3 for i in range(6):
4     if n % 2 == 1:
5         print(Names[i])
6     n //= 2

```

Задача II.1.3.2. Алфавитные подстроки (15 баллов)

Темы: задачи для начинающих, строки, реализация.

Условие

Алиса разрабатывает обучающую игру для младших школьников. В ней игроку дается строка из строчных символов латиницы, а он должен разбить ее на подстроки из последовательных символов алфавита. Такие подстроки далее будем называть правильными. В правильной подстроке после буквы a должна идти буква b , после b — c и так далее. При этом правильная подстрока может начинаться с любого символа. Например, строка $bcdefaabcef$ должна быть разбита на $bcdef+a+abc+ef$. Обратите внимание, что подстрока может состоять и из одного символа.

Конечно, игрок может ошибиться и разбить строку неправильным или неоптимальным способом. Например, игрок может разбить строку $bcdefaabcef$ на $bcd++ef+aabc+ef$. Чтобы учесть такую возможность Алиса считает очки за найденное разбиение. Если подстрока является правильной, то игроку добавляется количество очков, равное квадрату длины подстроки. Неправильные подстроки не учитываются. Например, за разбиение $bcdef+a+abc+ef$ игрок получит $5^2 + 1^2 +$

$+3^2 + 2^2 = 39$ очков, а за разбиение $bcd+ef+aabc+ef$ лишь $3^2 + 2^2 + 2^2 = 17$ очков.

Напишите программу, которая посчитает количество очков, полученных игроком за сделанное разбиение произвольной строки.

Формат входных данных

На вход в первой строке подается одно натуральное число n — количество фрагментов в разбиении, $1 \leq n \leq 100$. Далее записаны сами фрагменты разбиения. Каждый фрагмент записан в отдельной строке и состоит только из строчных символов латиницы. Длина каждого фрагмента не превосходит 26.

Формат выходных данных

Выведите одно целое число — количество очков, которое получит игрок за сделанное разбиение.

Методика проверки

Программа проверяется на 15-ти тестах. Прохождение каждого теста оценивается в 1 балл. Тест из условия задачи при проверке не используется. В первых четырех тестах разбиение состоит из одного фрагмента. В следующих четырех тестах каждый фрагмент содержит не более двух символов.

Примеры

Пример №1

Стандартный ввод
4 bcd ef aabc ef
Стандартный вывод
17

Пример программы-решения

Ниже представлено решение на языке Python 3.

```

1 n = int(input())
2 ans = 0
3 for i in range(n):
4     s = input()
5     for j in range(1, len(s)):
6         if ord(s[j]) - ord(s[j - 1]) != 1:
7             break

```

```
8     else:
9         ans += len(s) ** 2
10    print(ans)
```

Задача II.1.3.3. Пат и Паташон (15 баллов)

Темы: жадные алгоритмы, реализация.

Условие

Боб придумал логическую игру «Пат и Паташон». В этой игре на виртуальной сцене находится n персонажей различного роста, которые выстроены в один ряд. Игрок может удалить часть персонажей со сцены, при этом оставшиеся персонажи смыкаются, не изменяя своего взаимного расположения. После этого персонажи на сцене разбиваются на пары: первый со вторым, третий с четвертым, пятый с шестым и так далее.

В момент удаления игрок должен позаботиться о том, чтобы количество оставшихся персонажей стало четным. Первого персонажа в паре (с нечетным номером) будем называть Патом, а второго (с четным номером) — Паташоном. Эффектностью пары будем называть разность роста Пата и Паташона. Эффектность может быть отрицательной, если окажется, что Пат ниже, чем Паташон. За один раунд игрок получает количество очков, равное сумме эффектностей всех пар. Игрок может удалить со сцены всех персонажей. В этом случае он получит ноль очков.

Рассмотрим пример. Пусть на сцене изначально находилось девять персонажей, рост которых задается массивом чисел (120, 160, 180, 160, 120, 110, 150, 170, 100). Игрок удалил со сцены первого второго и седьмого персонажа. На сцене осталось шесть персонажей с ростом (180, 160, 120, 110, 170, 100). Они разбились на три пары (180, 160), (120, 110), (170, 100), при этом эффектность первой пары — 20, второй — 10, третьей — 70. Таким образом, за такое разбиение на пары игрок получит 100 очков. Однако, если игрок оставит на сцене четырех персонажей с ростом (180, 110, 170, 100), то он получит 140 очков.

Напишите программу, которая посчитает максимальное количество очков, которые может получить игрок, для заданной последовательности персонажей.

Формат входных данных

На вход в первой строке подается одно натуральное число n — количество персонажей на сцене в начале игры, $1 \leq n \leq 1000$. Далее в одной строке через пробел записана последовательность из n натуральных чисел, которые задают рост персонажей. Числа не превосходят 1000.

Формат выходных данных

Выведите одно целое число — максимальное количество очков, которое может получить игрок для заданной последовательности персонажей.

Методика проверки

Программа проверяется на 15-ти тестах. Прохождение каждого теста оценивается в 1 балл. Тест из условия задачи при проверке не используется. В первых пяти тестах на сцене изначально находится ровно четыре персонажа.

Примеры

Пример №1

Стандартный ввод
9 120 160 180 160 120 110 150 170 100
Стандартный вывод
140

Пример программы-решения

Ниже представлено решение на языке Python 3.

```
1 n = int(input())
2 x = list(map(int, input().split()))
3 print(sum([max(x[i]-x[i+1], 0) for i in range(n-1)]))
```

Задача II.1.3.4. Выезд на экскурсию (30 баллов)

Темы: математика, структуры данных, реализация.

Условие

Администрация школы организовала автобусную экскурсию для своих учеников. Всего было заказано n автобусов разной вместимости. Обозначим за c_i количество детей, которые могут находиться в i -том автобусе. Все c_i являются четными числами. Учителя неформально делят всех учеников на активных и спокойных. Всего на экскурсию поедет m активных и k спокойных детей. Учителя хотели бы распределить детей по автобусам так, чтобы количество спокойных и активных детей в каждом автобусе отличалось как можно меньше. Формально это означает следующее. Обозначим за x_i и y_i количество активных и спокойных детей соответственно в i -том автобусе. Вычислим модули разности количества активных и спокойных детей в каждом автобусе и просуммируем полученные числа. Полученная величина $\sum_{i=1}^n |x_i - y_i|$ должна оказаться минимально возможной.

Но когда автобусы подъехали, все пошло не по плану. Часть детей выбежали из школы и расселись по автобусам произвольно. После подсчетов выяснилось, что в i -том автобусе уже находится a_i активных детей и b_i спокойных. Чтобы не увеличивать неразбериху, было решено оставить их на своих местах и постараться рассадить оставшихся детей в соответствии с изначально выбранным принципом.

Напишите программу, которая найдет значения x_i и y_i с учетом всех требований, а именно:

- $x_i \geq a_i$;
- $y_i \geq b_i$;
- $x_i + y_i \leq c_i$;
- $\sum_{i=1}^n x_i = m$;
- $\sum_{i=1}^n y_i = k$;
- $\sum_{i=1}^n |x_i - y_i| \rightarrow \min$.

Если допустимых ответов несколько, то можно вывести любой.

Формат входных данных

На вход в первой строке через пробел подается три целых числа n , m и k — количество автобусов, количество активных и количество спокойных детей соответственно; $1 \leq n \leq 100$; $0 \leq m, k \leq 10000$. Во второй строке через пробел записаны числа a_1, a_2, \dots, a_n , задающие количество активных детей, изначально находящихся в каждом из автобусов. В третьей строке аналогично записаны числа b_1, b_2, \dots, b_n , задающие количество спокойных детей, изначально находящихся в каждом из автобусов. Наконец, в четвертой строке записаны натуральные четные числа c_1, c_2, \dots, c_n , задающие вместимость каждого из автобусов; $2 \leq c_i \leq 100$. Все входные значения заданы корректно в соответствии с условием задачи. В том числе гарантируется, что общее количество школьников не превосходит суммарной вместимости всех автобусов.

Формат выходных данных

Вывод должен состоять из двух строк. В первой строке через пробел следует вывести значения x_i — количество активных детей в каждом из автобусов. Во второй строке аналогично вывести значения y_i — количество спокойных детей в каждом из автобусов.

Методика проверки

Программа проверяется на 30-ти тестах. Прохождение каждого теста оценивается в 1 балл. Тест из условия задачи при проверке не используется. В первых пяти тестах количество автобусов равно двум. В следующих пяти тестах суммарное количество школьников равно суммарной вместимости автобусов.

Примеры

Пример №1

Стандартный ввод
4 50 80
10 20 0 0
25 5 20 0
40 30 40 30
Стандартный вывод
10 20 10 10
25 10 25 20

Пояснения к примеру

Ответ удовлетворяет всем ограничениям. Сумма всех x_i равна 50. Сумма всех y_i равна 80. В первом и третьем автобусе едет по 35 детей, а во втором и четвертом — по 30. Эти значения не превосходят вместимости соответствующих автобусов. Также для всех i выполняются неравенства $x_i \geq a_i$ и $y_i \geq b_i$. Значение выражения $\sum_{i=1}^n |x_i - y_i|$ равно 50. Можно доказать, что другие допустимые варианты распределения не дадут меньшей величины.

Возможны и другие правильные ответы, например, следующий.

```
15 20 15 0
25 10 20 25
```

Для этого ответа также выполнены все ограничения, а сумма $\sum_{i=1}^n |x_i - y_i|$ равна 50.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```
1 n, m, k = map(int, input().split())
2 a = list(map(int, input().split()))
3 b = list(map(int, input().split()))
4 c = list(map(int, input().split()))
5 m -= sum(a)
6 k -= sum(b)
7 for i in range(n):
8     if a[i] > b[i]:
9         v = min(k, a[i] - b[i], c[i] - a[i] - b[i])
10        b[i] += v
11        k -= v
12    else:
13        v = min(m, b[i] - a[i], c[i] - a[i] - b[i])
14        a[i] += v
15        m -= v
16 for i in range(n):
17    v = min(m, k, (c[i] - a[i] - b[i]) // 2)
18    a[i] += v
19    b[i] += v
20    m -= v
21    k -= v
22 for i in range(n):
23    v = min(m, c[i] - a[i] - b[i])
24    a[i] += v
25    m -= v
26    v = min(k, c[i] - a[i] - b[i])
27    b[i] += v
28    k -= v
29 print(*a)
30 print(*b)
```

Задача II.1.3.5. Нескучные каникулы (30 баллов)

Темы: сортировки, структуры данных, реализация.

Условие

У Алисы закончился очередной учебный год, и она составляет расписание на каникулы. Алиса планирует, что в ее каникулы состоится некоторое число событий, таких как посещение концертов, празднование дней рождений и так далее. Алиса называет i -тый день каникул нескучным, если для него выполняется хотя бы одно из двух условий:

- в i -тый день состоится хотя бы одно событие;
- хотя бы одно событие состоится в день с номером $i - 1$ и в день с номером $i + 1$.

Рассмотрим пример. Пусть в каникулах 10 дней и некоторые события произойдут в дни с номерами 2, 3, 5, 9, 10. Тогда нескучными будут все эти дни, а также день с номером 4, поскольку некоторые события произойдут в два соседних с ним дня.

При составлении расписания Алиса учитывает, что для некоторых событий заранее известна дата, а для других она сама может подобрать подходящий день. Алиса хочет расставить события с открытой датой так, чтобы каникулы получились наиболее нескучными, то есть чтобы количество нескучных дней в каникулах было максимальным.

Напишите программу, которая подберет дни для событий с открытой датой так, чтобы каникулы получились наиболее нескучными.

Формат входных данных

На вход в первой строке через пробел подается три целых числа n , m и k — продолжительность каникул в днях, количество событий с открытой датой и количество событий с заданной датой соответственно; $1 \leq n \leq 100000$; $1 \leq m \leq 100000$; $0 \leq k \leq 100000$.

Во второй строке через пробел записаны k натуральных чисел d_1, d_2, \dots, d_k — номера дней, в которые произойдут события с известной датой; $1 \leq d_i \leq n$. Числа могут повторяться и следовать в произвольном порядке. Если k будет равно нулю, то вторая строка будет пустой.

Формат выходных данных

В первой строке выведите одно натуральное число s — количество нескучных дней в каникулах. Во второй строке через пробел выведите m натуральных чисел t_1, \dots, t_m — номера дней, в которые Алиса должна запланировать события с открытой датой. Если допустимых ответов будет несколько, то можно вывести любой. Числа могут повторяться и следовать в произвольном порядке.

Методика проверки

Программа проверяется на 30-ти тестах. Прохождение каждого теста оценивается в 1 балл. Тесты из условия задачи при проверке не используются. В трех первых тестах $k = 0$. В следующих трех тестах $k = 1$. В первых 15-ти тестах n , m и k не превосходят 100.

*Примеры**Пример №1*

Стандартный ввод
11 5 6
1 3 5 7 9 11
Стандартный вывод
11
1 1 1 1 1

Пример №2

Стандартный ввод
11 2 0
Стандартный вывод
3
2 4

Пример №3

Стандартный ввод
15 2 5
1 2 8 12 14
Стандартный вывод
11
4 6

Пояснения к примеру

В первом примере все дни каникул являются нескучными из-за событий с известной датой, поэтому пять событий с открытой датой можно расставить произвольно.

В ответе ко второму примеру нескучными будут дни с номерами 2, 3, 4. Улучшить ответ нельзя.

В ответе к третьему примеру нескучными будут 11 дней с номерами 1, 2, 3, 4, 5, 6, 7, 8, 12, 13, 14. Улучшить этот ответ нельзя, хотя набор дней может быть другим, например, 4, 10 или 6, 10.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```

1 n, m, k = map(int, input().split())
2 d = [False] * (n + 2)
3 for x in map(int, input().split()):
4     d[x] = True
5 if k == 0:
6     ans = list(range(1, min(n, 2 * m), 2))

```

```
7     ans.extend([n] * max(m - len(ans), 0))
8 else:
9     p = 0
10    segs = []
11    for i in range(1, n + 1):
12        if d[i]:
13            if p == 0:
14                plen = i - 1
15            elif i - p > 2:
16                segs.append((p + 2, i))
17                p = i
18    segs.sort(key = lambda x: x[1] - x[0] + ((x[1] - x[0]) % 2) * 100000)
19    ans = []
20    for (a, b) in segs:
21        ans.extend(range(a, b, 2))
22    if n - p > 1:
23        ans.extend(range(p + 2, n + 1, 2))
24    if plen > 1:
25        ans.extend(range(plen - 1, 0, -2))
26    if (n - p) % 2 == 1:
27        ans.append(n)
28    ans = ans[: min(m, len(ans))]
29    ans.extend([1] * (m - len(ans)))
30    for i in ans:
31        d[i] = True
32    s = 0
33    for i in range(1, n + 1):
34        if d[i] or d[i - 1] and d[i + 1]:
35            s += 1
36    print(s)
37    print(*ans)
```

Предметный тур. Физика

Первая волна. Задачи 8–9 класса

Задача II.2.1.1. Конвейер (13 баллов)

Темы: кинематика.

Условие

Робот-доставщик по ошибке заехал на один из концов конвейерной ленты длиной L , движущуюся с постоянной скоростью v в противоположном ее движению направлению. Продолжая двигаться с постоянной относительно ленты скоростью u , робот сумел покинуть конвейер через время t . Определите u . Ответ дайте в м/с, округлив до сотых.

Решение

Скорость робота относительно земли $v_1 = L/t$ складывается (с правильным учетом знаков) из его скорости относительно поверхности ленты и скорости ленты относительно земли:

$$\frac{L}{t} = v_1 = u - v.$$

Откуда простыми алгебраическими преобразованиями получим:

$$u = v + \frac{L}{t}.$$

Погрешность 0,02 м/с.

Диапазоны

Величина	min	max	Шаг
L , м	19	23	1
v , м/с	1,6	1,9	0,05
t , с	150	250	10

Ответ: $u = v + \frac{L}{t}$.

Задача II.2.1.2. Выжигатель (17 баллов)

Темы: плотность, тепловые явления.

Условие

Температура кипения сплава на t выше его текущей температуры. Его удельная теплоемкость c , удельная теплота возгонки (испарения из твердого состояния) L , плотность ρ . Какой должна быть минимальная энергия E лазерного импульса, чтобы он был способен испарить кубик сплава со стороной a при условии полного поглощения энергии импульса веществом? Считайте, что импульс настолько кратковременный, что сплав не успевает пройти жидкую фазу и испаряется непосредственно из твердой. Ответ дайте в Дж, округлив до десятых.

Решение

Теплота, необходимая для нагрева и последующего испарения сплава массы m , находится по формуле:

$$Q = (ct + L)m.$$

По условиям вся энергия импульса поглощается веществом, то есть переходит в тепло. Следовательно, $E = Q$. Масса сплава может быть выражена через его плотность и объем испаренной порции: $m = \rho a^3$. Подставляя, получим:

$$E = (ct + L)\rho a^3.$$

Погрешность 0, 2 Дж.

Диапазоны

Величина	min	max	Шаг
t , °C	2200	2700	10
c , Дж/(кг·°C)	400	500	10
L , кДж/кг	5500	7000	100
ρ , кг/м ³	6000	8000	100
a , мм	0, 6	1	0, 1

Ответ: $E = (ct + L)\rho a^3$. С учетом порядков: $E = (ct + L[\cdot 10^3])\rho a^3[\cdot 10^{-9}]$.

Задача II.2.1.3. Катушка (20 баллов)

Темы: закон Ома.

Условие

При изготовлении реостата на диэлектрическую бобину диаметром D был намотан в N одинаковых, плотно прилегающих к бобине витков, константановый провод в лаковой изоляции. Площадь поперечного сечения провода S , удельное сопротивление константана $\rho = 0,4 \text{ Ом}\cdot\text{мм}^2/\text{м}$. Номинальное сопротивление резистора было вычислено и указано в паспорте устройства, исходя из этих параметров, однако от износа n последних витков проволоки перетерлось и отвалилось. Найдите максимальное значение n , при котором сопротивление реостата отличается от номинального не более, чем на ΔR .

Решение

Один виток провода имеет длину $l = \pi D$ и электрическое сопротивление:

$$r = \frac{\rho l}{S} = \frac{\pi \rho D}{S}.$$

Каждый отвалившийся виток уменьшает общее сопротивление резистора на величину r . Таким образом,

$$n = \left[\frac{\Delta R}{r} \right] = \left[\frac{\Delta R S}{\pi \rho D} \right].$$

Погрешность 1.

Диапазоны

Величина	min	max	Шаг
D , см	3	5	0,1
S , мм ²	0,1	0,2	0,02
N	100	250	10
ΔR , Ом	3	5	0,5

Ответ: $n = \left[\frac{\Delta R S}{\pi \rho D} \right]$. С учетом порядков: $n = \left[\frac{\Delta R S}{\pi \rho D} \cdot 10^2 \right]$.

Задача II.2.1.4. Болт (20 баллов)

Темы: золотое правило механики, работа.

Условие

При болтовом соединении деталей болт немного удлиняется, работая как растянутая пружина, прижимающая детали друг к другу с некоторой описанной в технической документации силой F . Определите, какую работу A нужно совершить, чтобы растянуть болт на величину Δl и создать таким образом силу F с помощью ключа длиной L , если диаметр резьбы болта равен d , диаметр шляпки D , шаг резьбы — b ? Трение и деформацию самих соединяемых деталей считайте пренебрежимо малым. Ответ дайте в Дж, округлив до целого.

Решение

Резьба болта и гаечный ключ представляют собой простые механизмы, дающие выигрыш в силе, но не изменяющие, согласно золотому правилу механики, работу. Поэтому общая работа, которая нужна, чтобы растянуть болт, вне зависимости от способа, считается как работа переменной силы или как разность потенциальных энергий деформированного тела (болта). При этом малость деформации позволяет применить здесь закон Гука:

$$A = \frac{F \Delta l_{\text{к}}}{2} - \frac{F \Delta l_{\text{н}}}{2}.$$

Учитывая $\Delta l_{\text{н}} = 0$, получим окончательно:

$$A = \frac{F\Delta l}{2}.$$

Погрешность 1 Дж.

Диапазоны

Величина	min	max	Шаг
F , кН	200	250	10
Δl , мм	0,25	0,35	0,01
L , см	40	70	5
d , мм	20	25	1
D , мм	30	40	1
b , мм	2	3	0,1

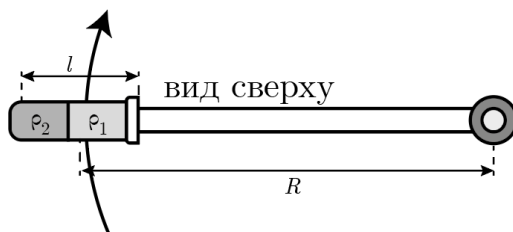
Ответ: $A = \frac{F\Delta l}{2}$.

Задача II.2.1.5. Центрифуга (25 баллов)

Темы: центростремительное ускорение, гидростатическое давление.

Условие

Для разделения жидких смесей в лаборатории используется центрифуга, представляющая собой горизонтальное колесо радиуса R , на ободе которого закрепляются пробирки с жидкостью, ориентированные дном строго от центра колеса. Найдите давление, оказываемое на дно пробирки длиной l при ее вращении в центрифуге, если пробирка совершает один оборот за время T и ровно на половину своего объема пробирка заполнена составом с плотностью ρ_1 и ровно на половину — другим составом с плотностью ρ_2 . Считайте $R \gg l$, а влияние силы тяжести пренебрежимо малым. Ответ дайте в кПа, округлив до целого. Длина окружности в 2π раз больше ее радиуса.



Решение

Линейная скорость пробирки может быть найдена по формуле $v = 2\pi R/T$, поскольку путь, проходимый пробиркой за время T , равен длине окружности с радиусом R . Соответствующее ей центростремительное ускорение a равно:

$$a = \frac{v^2}{R} = \frac{4\pi^2 R}{T^2}.$$

Это ускорение играет роль ускорения свободного падения в формуле гидростатического давления ρgh , что может быть доказано из условий равновесия элемента жидкости. Поскольку жидкости в пробирке две и каждая из них создает столб «высотой» $l/2$, для общего давления получим:

$$p = \frac{2\pi^2 Rl}{T^2} (\rho_1 + \rho_2).$$

Погрешность 3 кПа.

Диапазоны

Величина	min	max	Шаг
ν , об/с	5	7	0,5
R , см	90	120	5
l , см	6	8	1
ρ_1 , кг/м ²	800	980	20
ρ_2 , кг/м ²	1020	1200	20

Ответ: $p = \frac{2\pi^2 Rl}{T^2} (\rho_1 + \rho_2)$. С учетом порядков $p = \frac{2\pi^2 Rl}{T^2} (\rho_1 + \rho_2) [\cdot 10^{-7}]$.

Задача II.2.1.6. (5 баллов)

Темы: физики России.

Условие

Этого выдающегося ученого, обучавшегося еще в технологическом институте Николая I, но удостоенного и ленинской, и сталинской премий, нередко называют родоначальником советской физики. Такие известные физики как Капица и Курчатов достигли своих выдающихся результатов под его руководством, а сам он был воспитан под руководством первооткрывателя «икс-лучей», используемых теперь в каждой поликлинике.

1. Петр Николаевич Лебедев.
2. Абрам Федорович Иоффе.
3. Николай Алексеевич Умов.
4. Александр Александрович Фридман.
5. Сергей Александрович Ахманов.

6. Рем Викторович Хохлов.
7. Владимир Александрович Фок.
8. Михаил Васильевич Остроградский.

Ответ: 2.

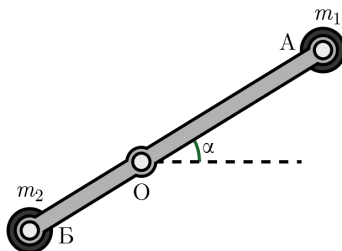
Первая волна. Задачи 10–11 класса

Задача II.2.2.1. Противовес (15 баллов)

Темы: кинематика, импульс.

Условие

Легкий рычаг АОБ, способный вращаться вокруг неподвижной точки О, где $BO = l$, $OA = 2l$ используется на производственной линии для перемещения грузов. В его точке А закреплен груз m_1 , в точке Б — противовес m_2 . В некоторый момент модуль импульса груза в точке А равен p_1 . Найдите модуль импульса противовеса в этот же момент времени. Дайте ответ в $\text{кг} \cdot \text{м/с}$, округлив до целого.



Решение

Поскольку $BO : OA = 1 : 2$, скорости двух масс всегда связаны соотношением $v_1 = 2v_2$. Тогда их импульсы связаны соотношением:

$$\frac{p_2}{p_1} = \frac{m_2 v_2}{m_1 v_1} = \frac{m_2}{2m_1}.$$

Отсюда окончательно получим:

$$p_2 = p_1 \frac{m_2}{2m_1}.$$

Погрешность $1 \text{ кг} \cdot \text{м/с}$.

Диапазоны

Величина	min	max	Шаг
l , м	1	1,5	0,1
m_1 , кг	50	80	5
m_2 , кг	50	80	5
p_1 , кг · м/с	50	160	10

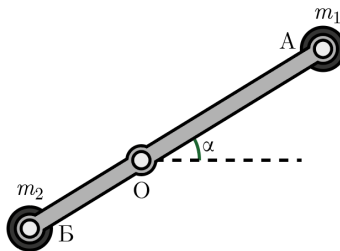
Ответ: $p_2 = p_1 \frac{m_2}{2m_1}$.

Задача II.2.2.2. Качели (20 баллов)

Темы: законы сохранения.

Условие

Легкий рычаг АОБ, способный вращаться вокруг неподвижной точки О, где $BO = l$, $OA = 2l$ используется, на производственной линии для перемещения грузов. В его точке А закреплен груз m_1 , в точке Б — противовес m_2 . В некоторый момент электропривод в точке О удерживал рычаг под углом α к горизонту. В этот момент произошла авария, в результате которой привод в точке О перестал действовать, и рычаг пришел в свободное вращение вокруг этой оси. Определите скорость v_1 точки А рычага в момент, когда она оказалась строго под точкой Б. Дайте ответ в м/с, округлив до десятых. Ускорение свободного падения $g = 9,8 \text{ м/с}^2$.



Решение

Поскольку $BO : OA = 1 : 2$, скорости двух масс всегда связаны соотношением $v_1 = 2v_2$. Высоты (относительно точки О), на которых находились массы в момент аварии, равны $h_{01} = 2l \sin \alpha$, $h_{02} = -l \sin \alpha$. Аналогично высоты, на которых находились массы в момент, когда точка Б оказывается строго под точкой А, равны $h_1 = -2l$, $h_2 = l$.

Тогда закон сохранения энергии для масс на концах рычага имеет вид:

$$(2m_1 - m_2)gl \sin \alpha = (m_2 - 2m_1)gl + \left(m_1 + \frac{m_2}{4}\right) \frac{v_1^2}{2}.$$

Из этого уравнения можно непосредственно выразить скорость v_1 :

$$v_1 = \sqrt{\frac{8gl(2m_1 - m_2)(1 + \sin \alpha)}{4m_1 + m_2}}.$$

Погрешность 0, 2 км/ч.

Диапазоны

Величина	min	max	Шаг
l , м	1	1, 5	0, 1
m_1 , кг	50	80	5
m_2 , кг	50	80	5
α , °	30	70	5

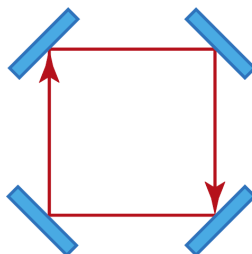
Ответ: $v_1 = \sqrt{\frac{8gl(2m_1 - m_2)(1 + \sin \alpha)}{4m_1 + m_2}}.$

Задача II.2.2.3. Резонатор (20 баллов)

Темы: оптика.

Условие

Кольцевой оптический резонатор представляет собой квадрат со стороной a , в углах которого расположены наклоненные под 45° к ходу луча зеркала. При каждом падении на зеркало $k\%$ энергии света отражается, а остальные $(100 - k)\%$ поглощаются материалом зеркала. Какая доля изначальной энергии останется у светового луча, когда он сделает N полных витков? Ответ дайте в процентах.



Решение

За один виток свет совершает 4 отражения. В ходе каждого из них его энергия изменяется в $k/(100\%)$ раз. Таким образом, общая η доля сохранившейся энергии равна:

$$\eta = \left(\frac{k}{100\%} \right)^{4N} \cdot 100\%.$$

Погрешность 1%.

Диапазоны

Величина	min	max	Шаг
a , см	10	50	5
k	98,5	99,5	0,5
N	20	30	1

Ответ: $\eta = \left(\frac{k}{100\%} \right)^{4N} \cdot 100\%.$

Задача II.2.2.4. Спидометр (20 баллов)

Темы: кинематика.

Условие

Спидометр спутниковой системы навигации, показывая текущую скорость, на самом деле определяет ее как среднюю скорость за последние t движения. Определите мгновенную скорость аппарата, если его спидометр сейчас показывает скорость v_1 , t назад показывал скорость v_2 и известно, что аппарат движется равноускоренно. Ответ дайте в км/ч, округлив до десятых.

Решение

При равноускоренном движении на протяжении некоторого времени t с начальной скоростью v_0 и ускорением a конечная скорость оказывается равна $v = v_0 + at$, а средняя

$$v_{\text{ср}} = \frac{v_0 + v}{2} = v_0 + \frac{at}{2}.$$

При этом, поскольку по прошествии дополнительного времени t и v , и v_0 вырастет на at , средняя скорость также вырастет на at . Следовательно, ускорение аппарата может быть найдено как

$$a = \frac{v_1 - v_2}{t},$$

а его мгновенная скорость:

$$v = v_1 + \frac{at}{2} = v_1 + \frac{v_1 - v_2}{2} = \frac{3v_1 - v_2}{2}.$$

Погрешность 0,1 км/ч.

Диапазоны

Величина	min	max	Шаг
t , с	1,5	3	0,25
v_1 , км/ч	52	56	0,5
v_2 , км/ч	47	51	0,5

Ответ: $v = \frac{3v_1 - v_2}{2}$.

Задача II.2.2.5. Разрядка (20 баллов)

Темы: электростатика.

Условие

Испытываемая в лаборатории антенна состоит из двух одинаковых металлических шаров, разнесенных на некоторое расстояние и соединенных перемычкой из материала с очень высоким, но конечным сопротивлением. В результате шары обмениваются зарядом таким образом, что за каждый интервал времени t разница между их зарядами сокращается вдвое. В некоторый момент первый шар был заряжен, а второй — нет. Спустя t было измерено, что сила электростатического взаимодействия между шарами антенны равна F_0 . Какой станет эта сила, спустя еще t времени? Явлением электростатической индукции пренебречь. Ответ дайте в нН (наноньютоны), округлив до целого.

Решение

Пусть в начальный момент времени заряженный шар имеет заряд q . Тогда, спустя t , заряды на шарах должны отличаться на $q/2$, что, по закону сохранения заряда, возможно только если их заряды будут равны $3q/4$ и $q/4$. Соответствующая сила взаимодействия находится по закону Кулона:

$$F_0 = k \frac{(q/4)(3q/4)}{r^2} = \frac{3}{16} \cdot \frac{kq^2}{r^2}.$$

Спустя еще t разница между зарядами должна оказаться равна $q/4$, а заряды, соответственно, $5q/8$ и $3q/8$. Соответствующая сила:

$$F = k \frac{(3q/8)(5q/8)}{r^2} = \frac{15}{64} \cdot \frac{kq^2}{r^2} = \frac{5}{4} F_0.$$

Погрешность 1 нН.

Диапазоны

Величина	min	max	Шаг
t , с	5	40	1
F , нН	10	100	2

Ответ: $F = \frac{5}{4} F_0$.

Задача II.2.2.6. (5 баллов)

Темы: физики России.

Условие

Этого выдающегося ученого, обучавшегося еще в технологическом институте Николая I, но удостоенного и ленинской, и сталинской премий, нередко называют родоначальником советской физики. Такие известные физики как Капица и Курчатов достигли своих выдающихся результатов под его руководством, а сам он был воспитан под руководством первооткрывателя «икс-лучей», используемых теперь в каждой поликлинике.

1. Рем Викторович Хохлов.
2. Абрам Федорович Иоффе.
3. Владимир Александрович Фок.
4. Александр Александрович Фридман.
5. Сергей Александрович Ахманов.
6. Петр Николаевич Лебедев.
7. Николай Алексеевич Умов.
8. Михаил Васильевич Остроградский.

Ответ: 2.

Вторая волна. Задачи 8–9 класса**Задача II.2.3.1. Катапульта (10 баллов)**

Темы: кинематика.

Условие

Для запуска беспилотного летательного аппарата используется катапульта — длинная балка с установленным на ней линейным двигателем, способным создавать постоянное ускорение a . Для успешного взлета аппарат должен успеть набрать скорость v относительно воздуха до отрыва от катапульты. Вычислите минимальную необходимую длину катапульты, если взлет должен успешно осуществляться при попутном ветре со скоростью не выше u . Ответ дайте в м, округлив до десятых.

Решение

При попутном ветре скорость аппарата относительно воздуха равна $v = v_0 + u$, где v_0 — его скорость относительно катапульты. Как известно из кинематики (или законов сохранения), на расстоянии l , двигаясь с постоянным ускорением a , беспилотник может увеличить квадрат скорости на величину $v^2 = 2al_0$. Отсюда найдем изначальную длину балки:

$$l = \frac{v^2}{2a} = \frac{(v + u)^2}{2a}.$$

Погрешность 0,1 м.

Диапазоны

Величина	min	max	Шаг
a , м/с ²	25	40	1
v , м/с	9	12	0,5
u , м/с	2	4	0,5

Ответ: $l = \frac{(v + u)^2}{2a}$.

Задача II.2.3.2. Отвердевание (15 баллов)

Темы: плотность.

Условие

Некоторый полимер был получен в жидком состоянии, в котором он целиком занимал лабораторную чашку объемом V_0 и имел плотность ρ_0 . Затем он был нагрет в специальной печи, в результате чего из состава испарились все летучие фракции общей массой Δm , а оставшееся вещество застыло, образовав твердый сгусток с плотностью ρ . Определите объем, занимаемый твердым веществом в чашке, если известно, что пустоты в процессе застывания не образуются и никакие внешние соединения не включаются в полимер. Ответ дайте в см³, округлив до целого.

Решение

Изначальная масса полимера равна $m_0 = \rho_0 V_0$. За счет испарения она уменьшилась на Δm . Процесс отвердевания не изменяет массы вещества, поэтому объем образовавшегося твердого вещества равен $V = (m_0 - \Delta m) / \rho$. Окончательно:

$$V = \frac{\rho_0 V_0 - \Delta m}{\rho}.$$

Погрешность 5 см³.

Диапазоны

Величина	min	max	Шаг
V_0 , см ³	1200	1500	100
ρ_0 , г/см ³	0,9	1,1	0,02
ρ , г/см ³	1,4	1,7	0,02
Δm , г	100	150	10

Ответ: $V = \frac{\rho_0 V_0 - \Delta m}{\rho}$.

Задача II.2.3.3. Три образца (20 баллов)

Темы: электростатика.

Условие

В лабораторию поступило три металлических образца одинаковых размеров А, Б и В, электрически заряженных в ходе трех разных малоисследованных процессов. Лаборанты записали результаты измерений их зарядов $q_{1,2,3}$, но позже выяснилось, что процедура проведения эксперимента была нарушена и было допущено соприкосновение образцов друг с другом. В ходе анализа записей лабораторной камеры удалось установить, что вначале соприкоснулись образцы А и Б, после чего на одном из них был измерен заряд q_1 . Затем с этим образцом дополнительно соприкоснулся образец В, после чего на нем был измерен заряд q_2 . После этого все три образца соприкоснулись одновременно и на одном из них был измерен заряд q_3 . Восстановите по этим данным исходное значение заряда образца В. Считайте, что процедура измерения величины заряда на образце не меняет. Ответ дайте в нКл (нанокулонах), округлив до целого.

Решение

Когда два металлических предмета одинаковых размеров соприкасаются, на них устанавливаются одинаковые электрические заряды. По закону сохранения заряда они должны быть равны среднему арифметическому исходных зарядов.

Тогда после первого соприкосновения

$$q_1 = \frac{q_A + q_B}{2},$$

после второго

$$q_2 = \frac{q_1 + q_B}{2},$$

а после третьего

$$q_3 = \frac{q_A + q_B + q_B}{3} = \frac{2q_1 + q_B}{3}.$$

Решая эту систему уравнений, легко получим:

$$q_B = 4q_2 - 3q_3.$$

Погрешность 1 нКл.

Диапазоны

Величина	min	max	Шаг
q_1 , нКл	-20	20	4
q_2 , нКл	-20	20	4
q_3 , нКл	-20	20	4

Ответ: $q_B = 4q_2 - 3q_3$.

Задача II.2.3.4. Топливо (25 баллов)

Темы: тепловые явления.

Условие

На некоторой планете были обнаружены залежи жидкого состава, состоящего из двух трудно разделяемых компонент. Первая из них химически инертна (не участвует ни в каких превращениях) и имеет теплоемкость c_1 . Вторая имеет теплоемкость c_2 и горит в атмосфере планеты с удельной теплотой сгорания q . Смесь воспламеняется при температуре θ , а окружающая среда планеты имеет температуру t . Определите, какую минимальную долю от общей массы смеси должна составлять масса горючей компоненты, чтобы горение смеси производило не меньше энергии, чем уходит на ее нагрев до температуры воспламенения. Ответ дайте в процентах, округлив до десятых.

Решение

Рассмотрим некоторую массу m смеси. Она содержит $m_2 = \alpha m$ горючей и $m_1 = (1 - \alpha)m$ негорючей жидкостей. Для воспламенения необходимо нагреть смесь на $\Delta t = \theta - t$, на что уйдет

$$Q_1 = (c_1 m_1 + c_2 m_2) \Delta t = m(c_1(1 - \alpha) + c_2 \alpha)(\theta - t)$$

теплоты. При этом от сгорания второй компоненты выделится $Q_2 = q m_2$ теплоты. Приравнявая Q_1 и Q_2 , получим:

$$m \alpha q = m(c_1(1 - \alpha) + c_2 \alpha)(\theta - t),$$

откуда окончательно выразим ответ:

$$\alpha = \frac{c_1(\theta - t)}{(\theta - t)(c_1 - c_2) + q} \cdot 100\%.$$

Погрешность 0,1%.

Диапазоны

Величина	min	max	Шаг
c_1 , Дж/(кг·°C)	300	500	20
c_2 , Дж/(кг·°C)	310	510	20
q , МДж/(кг)	13	17	0,5
θ , °C	800	950	10
t , °C	-80	-50	5

Ответ: $\alpha = \frac{c_1(\theta - t)}{(\theta - t)(c_1 - c_2) + q} \cdot 100\%$.

С учетом порядков $\alpha = \frac{c_1(\theta - t)}{(\theta - t)(c_1 - c_2) + q \cdot [10^6]} \cdot 100\%$.

Задача II.2.3.5. Геккон (25 баллов)

Темы: давление, сила трения.

Условие

Робот-геккон может перемещаться по гладким вертикальным поверхностям, используя присоски. Под каждой присоской площади S при помощи системы компрессоров, откачивающих из-под присосок воздух, создается давление p , в то время как атмосферное давление $p_0 = 100$ кПа.

Какую максимальную массу может иметь такой робот, чтобы не соскальзывать с вертикальной поверхности, если он одновременно использует n одинаковых присосок (коэффициент трения резины присосок равен μ)?

Ускорение свободного падения принять равным $g = 9,8$ м/с². Ответ дайте в кг, округлив до десятых.

Решение

На каждую присоску действует прижимная сила, равная произведению ее площади на разницу внешнего и внутреннего давлений:

$$N = S(p_0 - p).$$

Возникающие в этих присосках силы трения $F_{\text{тр}} = \mu N$ должны суммарно уравновешивать вес робота:

$$mg = nF_{\text{тр}} = \mu nS(p_0 - p).$$

Таким образом, окончательно

$$m = \frac{\mu nS(p_0 - p)}{g}.$$

Погрешность 0,2 кг.

Диапазоны

Величина	min	max	Шаг
S , см ²	10	35	2, 5
μ	0, 12	0, 25	0, 01
p , кПа	10	40	5
n	6	12	2

Ответ: $m = \frac{\mu nS(p_0 - p)}{g}$. С учетом порядков $m = \frac{\mu nS(p_0 - p)}{g} \cdot [10^{-1}]$.

Задача II.2.3.6. (5 баллов)

Темы: физики России.

Условие

Один из основателей нелинейной оптики, новой ветви науки, продемонстрировавшей, что достаточно интенсивные лучи света могут взаимодействовать друг с другом и сами с собой, фокусироваться без линзы и неожиданно менять цвет. В его честь на территории Московского университета названа улица, лаборатория, спортивный клуб и несколько учебных аудиторий. Помимо выдающихся научных достижений, он также проявил себя как талантливый организатор, способствовал развитию кооперации исследователей самых разных направлений, включая биологию и экологию, а также был профессиональным альпинистом с двадцатилетним стажем. Трагедия в одном из горных походов, к несчастью, оборвала его выдающуюся жизнь.

1. Сергей Александрович Ахманов.
2. Рем Викторович Хохлов.
3. Анатолий Алексеевич Логунов.
4. Петр Николаевич Лебедев.
5. Абрам Федорович Иоффе.
6. Александр Александрович Фридман.
7. Роберт Эмильевич Ленц.
8. Николай Алексеевич Умов.

Ответ: 2.

Вторая волна. Задачи 10–11 класса**Задача II.2.4.1. Кабель (12 баллов)**

Темы: закон Ома, сопротивление.

Условие

На мобильной исследовательской станции используются стандартные резервные кабели для большинства электроприборов, имеющие площадь поперечного сечения s и длину l . В документации кабеля указано, что при подключении к стандартному лабораторному источнику постоянного тока I падение напряжения на кабеле составляет U . Определите удельное сопротивление материала кабеля. Ответ дайте в Ом · мм²/м, округлив до тысячных.

Решение

Сопротивления r кабеля питания вычисляются по формуле:

$$r = \frac{\rho l}{s}.$$

Согласно закону Ома для участка цепи $I = U/r$, оно также может быть выражено в виде:

$$\frac{\rho l}{s} = r = \frac{U}{I}.$$

Отсюда окончательно получим:

$$\rho = \frac{U_s}{Il}.$$

Погрешность $0,002 \text{ Ом}\cdot\text{мм}^2/\text{м}$.

Диапазоны

Величина	min	max	Шаг
l , м	5	8	0,5
s , мм ²	1,6	2	0,1
I , А	1	2	0,1
U , В	0,1	0,2	0,01

Ответ: $\rho = \frac{U_s}{Il}$.

Задача II.2.4.2. Мороз (15 баллов)

Темы: закон Джоуля – Ленца.

Условие

На мобильной полярной станции вышла из строя основная система обогрева, и пришлось в срочном порядке подключать давно не использовавшуюся резервную. Резервный нагреватель представляет собой теплопроводящий корпус, защищающий катушку, на которую намотан провод длиной L и площадью поперечного сечения S , обеспечивающий эффективную конвекцию. Штатный кабель питания от нагревателя, к сожалению, был утерян, поэтому нагреватель пришлось подключить к сети при помощи стандартного резервного кабеля, имеющего площадь s и длину l . Определите отношение P_n/P_k тепловой мощности, выделяющейся в нагревателе к тепловой мощности, выделяющейся в кабеле питания, если токонесущие жилы кабеля питания и провода нагревателя изготовлены из одного вещества. Дайте ответ с точностью до сотых.

Решение

Сопротивления R нагревателя и r кабеля питания вычисляются по формулам:

$$R = \rho \frac{L}{S}; \quad r = \rho \frac{l}{s},$$

где ρ – (одинаковое в обоих случаях) удельное сопротивление. Их отношение равно

$$\frac{R}{r} = \frac{Ls}{lS}.$$

Поскольку кабель и нагреватель включены в цепь последовательно, проходящие через них силы тока равны и тепловую мощность удобно искать по закону Джоуля – Ленца:

$$P_n = I^2 R; \quad P_k = I^2 r.$$

Таким образом, окончательно

$$\frac{P_n}{P_k} = \frac{I^2 R}{I^2 r} = \frac{Ls}{lS}.$$

Погрешность 0,05.

Диапазоны

Величина	min	max	Шаг
L , м	15	20	1
l , м	5	8	0,5
S , мм ²	2,5	3,5	0,2
s , мм ²	1,6	2	0,1

Ответ: $\frac{P_n}{P_k} = \frac{Ls}{lS}.$

Задача II.2.4.3. Стопка (20 баллов)

Темы: конденсаторы.

Условие

На тонкий полимерный лист с обеих сторон напыляется металлическое покрытие. Измерения показывают, что емкость такого конденсатора равна C_0 . Затем лист разрезают на n равных частей и складывают их в стопку. Определите емкость C такой стопки при подключении источника напряжения между самым верхним и самым нижним ее слоями. Ответ дайте в пФ (пикофарадах), округлив до целого.

Решение

Емкость плоского конденсатора задается формулой:

$$C = \frac{\varepsilon\varepsilon_0 S}{d},$$

поэтому при разрезании листа на n частей площадь и, соответственно, емкость C_1 каждой из этих частей, оказывается в n раз меньше исходной: $C_1 = C_0/n$.

В то же время стопка конденсаторов представляет собой цепь из n одинаковых конденсаторов, соединенных последовательно. Поскольку эффективная емкость C такой цепи может быть найдена по формуле:

$$\frac{1}{C} = \frac{1}{C_1} + \frac{1}{C_1} + \dots + \frac{1}{C_1} = \frac{n}{C_1} = \frac{n^2}{C_0},$$

получим окончательно

$$C = \frac{C_0}{n^2}.$$

Погрешность 2 пФ.

Диапазоны

Величина	min	max	Шаг
C , нФ	2	6	0,5
n	4	10	1

Ответ: $C = \frac{C_0}{n^2}$. С учетом порядков $C = \frac{C_0}{n^2} [\cdot 10^3]$.

Задача II.2.4.4. Два колеса (23 баллов)

Темы: кинematика.

Условие

Два колеса двухколесного балансирующего робота, исследующего отдаленный астрономический объект с очень гладкой поверхностью, расположены на разных концах одной оси длиной l . Проведя сеанс связи с Землей, робот получил указание двигаться прямо и стал вращать колеса с одинаковой угловой скоростью. Однако в ходе посадки робота одно из колес было слегка повреждено, в связи с чем его радиус оказался на Δr меньше, чем радиус r другого. Найдите Δr , если известно, что вместо прямой робот начал двигаться по окружности радиуса R (измеренного по центру робота). Колеса робота не проскальзывают по поверхности, кривизна астрономического объекта пренебрежимо мала. Ответ дайте в мм, округлив до десятых.

Решение

При равных угловых скоростях ω линейные скорости колес имеют величины ωr и $\omega(r - \Delta r)$. При этом, поскольку робот движется по окружности, то за время, за которое внешнее колесо проходит дугу длиной $\alpha(R + l/2)$, внутреннее колесо проходит дугу $\alpha(R - l/2)$. Следовательно

$$\frac{\omega(r - \Delta r)}{\omega r} = \frac{R - l/2}{R + l/2}.$$

Преобразуя это выражение, получим окончательно:

$$\Delta r = \frac{2rl}{2R + l}.$$

Погрешность 0,2 мм.

Диапазоны

Величина	min	max	Шаг
r , см	40	60	4
l , см	80	120	10
R , м	80	120	5

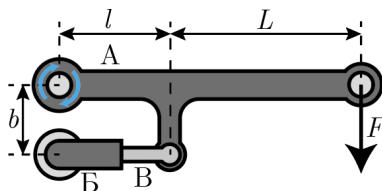
Ответ: $\Delta r = \frac{2rl}{2R+l}$. С учетом порядков $\Delta r = \frac{2rl \cdot [10^4]}{2R \cdot [10^2] + l}$.

Задача II.2.4.5. Рычаг (25 баллов)

Темы: статика, давление газа.

Условие

Подъемный рычаг А, геометрические параметры b, l, L которого изображены на рисунке, шарнирно соединен с приводящим его в движение поршнем В, входящим в цилиндр Б, внутри которого содержится идеальный газ. Пренебрегая весом самого рычага, определите избыточное (над атмосферным) давление p в цилиндре, которое необходимо для удержания нагрузки F , приложенной к концу рычага, если площадь поршня равна S . Ответ дайте в МПа, округлив до десятых.



Решение

Относительно шарнира рычага сила F имеет плечо $L+l$. Сила давления поршня равна pS и имеет плечо b . Тогда условие равновесия имеет вид:

$$bpS = (l + L)F.$$

Отсюда

$$p = \frac{(l + L)F}{bS}.$$

Погрешность 0,2 МПа.

Диапазоны

Величина	min	max	Шаг
l , см	15	25	2
L , см	30	50	5
b , см	8	16	1
S , см ²	16	24	2
F , кН	2	3	0,5

Ответ: $p = \frac{(l + L)F}{bS}$. С учетом порядков $p = \frac{(l + L)F}{bS} \cdot [10^4]$.

Задача II.2.4.6. (5 баллов)

Темы: физики России.

Условие

Один из основателей нелинейной оптики, новой ветви науки, продемонстрировавшей, что достаточно интенсивные лучи света могут взаимодействовать друг с другом и сами с собой, фокусироваться без линзы и неожиданно менять цвет. В его честь на территории Московского университета названа улица, лаборатория, спортивный клуб и несколько учебных аудиторий. Помимо выдающихся научных достижений, он также проявил себя как талантливый организатор, способствовал развитию кооперации исследователей самых разных направлений, включая биологию и экологию, а также был профессиональным альпинистом с двадцатилетним стажем. Трагедия в одном из горных походов, оборвала его выдающуюся жизнь.

1. Сергей Александрович Ахманов.
2. Рем Викторович Хохлов.
3. Анатолий Алексеевич Логунов.
4. Петр Николаевич Лебедев.
5. Александр Александрович Фридман.
6. Николай Алексеевич Умов.
7. Абрам Федорович Иоффе.
8. Роберт Эмильевич Ленц

Ответ: 2.

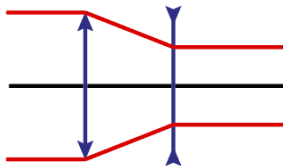
Третья волна. Задачи 8–9 класса**Задача II.2.5.1. Большой глаз (15 баллов)**

Темы: геометрическая оптика.

Условие

Первый элемент системы ночного наблюдения имеет своей целью сузить пучок параллельных лучей, собираемых с большой площади, оставив его параллельным и неперевернутым. Он состоит из двух линз: собирающей и рассеивающей с оптическими силами D_1 и D_2 соответственно, установленных друг за другом и имеющих общую оптическую ось. Найдите расстояние между линзами. Ответ дайте в см, округлив до десятых.

Напоминание: оптической силой называется величина, обратная фокусному расстоянию линзы.



Решение

Изобразим описанную в условиях задачи оптическую схему. Из рисунка несложно видеть, что для описанного хода лучей фокусы двух линз должны совпадать. Но по определению оптической силы, фокусное расстояние линзы равно

$$F = \frac{1}{D}.$$

Подставляя в эту формулу D_1, D_2 и находя разницу между соответствующими расстояниями (верно учитывая знаки), получим:

$$l = \frac{1}{D_1} + \frac{1}{D_2}.$$

Погрешность 0,2 см.

Диапазоны

Величина	min	max	Шаг
D_1 , дптр	3	5	0,2
D_2 , дптр	-10	-8	0,2

Ответ: $l = \frac{1}{D_1} + \frac{1}{D_2}$. С учетом порядков $l = \left(\frac{1}{D_1} + \frac{1}{D_2} \right) [\cdot 10^2]$.

Задача II.2.5.2. Атмосфера (15 баллов)

Темы: давление газа, гидростатика.

Условие

Оцените массу атмосферы, окружающей планету земного типа радиусом R , если ускорение свободного падения на ее поверхности равно g , а атмосферное давление — p_0 . Площадь сферы вычисляется по формуле $S = 4\pi R^2$. Ответ дайте в квинтлн (квинтиллионах) (10^{18}) кг, округлив до десятых.

Решение

Давление — это отношение силы к площади, перпендикулярно которой эта сила действует. В данном случае сила — общий вес атмосферы:

$$p_0 = \frac{mg}{S} = \frac{mg}{4\pi R^2}.$$

Для любой планеты земного типа толщина атмосферы пренебрежимо мала в сравнении с радиусом планеты, поэтому изменением ускорения свободного падения с высотой можно пренебречь. Получим окончательно

$$m = \frac{4\pi R^2 p_0}{g}.$$

Погрешность $0,1 \cdot 10^{15}$ т.

Диапазоны

Величина	min	max	Шаг
p_0 , кПа	12	24	1
g , м/с ²	2	3,5	0,1
R , км	3600	4600	100

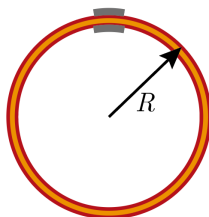
Ответ: $m = \frac{4\pi R^2 p_0}{g}$. С учетом порядков $m = \frac{4\pi R^2 p_0}{g} [\cdot 10^{-9}]$.

Задача II.2.5.3. Магниты (20 баллов)

Темы: центростремительное ускорение, динамика.

Условие

К тонкому ободу медного колеса радиусом R , расположенного в горизонтальной плоскости, снаружи и изнутри прикреплены два одинаковых маленьких магнитных датчика массой m каждый. Датчики держатся только за счет притяжения друг к другу. Колесо начинают постепенно раскручивать вокруг неподвижной оси, и в момент, когда период его вращения достигает величины T , внешний датчик отлетает от колеса. Определите силу давления внутреннего датчика на обод колеса непосредственно после этого. Ответ дайте в Н, округлив до целого. Длина окружности в 2π раз больше ее радиуса.



Решение

Внешний датчик отделяется в момент, когда сила F взаимодействия между магнитами оказывается недостаточна для того, чтобы создавать центростремительное ускорение $a = v^2/R$, где скорость v может быть найдена как отношение длины окружности к периоду обращения ($v = 2\pi R/T$):

$$F = ma = \frac{mv^2}{R} = \frac{4\pi^2}{T^2}mR.$$

Поскольку размеры датчиков и толщина колеса считаются пренебрежимо малыми, внутренний датчик продолжает двигаться с тем же ускорением, а значит, давить на колесо с той же силой F :

$$F = \frac{4\pi^2}{T^2}mR.$$

Погрешность 1 Н.

Диапазоны

Величина	min	max	Шаг
R , см	36	45	3
m , г	200	300	20
T , с	0,3	0,5	0,05

Ответ: $F = \frac{4\pi^2}{T^2}mR$. С учетом порядков $F = \frac{4\pi^2}{T^2}mR \cdot 10^{-5}$.

Задача II.2.5.4. Пот (20 баллов)

Темы: тепловые явления, кинематика жидкости.

Условие

Для охлаждения антропоморфного робота разрабатывается система, воспроизводящая потоотделение человека. По тонким капиллярам с площадью поперечного сечения S на поверхность робота поступает вода, которая затем растекается по «коже» робота и постепенно испаряется. Определите, какой должна быть постоянная скорость v течения воды в капилляре, если один такой капилляр должен обеспечивать отведение тепловой мощности N . Считайте, что вся вода успевает испариться. Удельная теплота парообразования воды $L = 2300$ кДж/кг, ее плотность $\rho = 1000$ кг/м³. Ответ дайте в мм/с, округлив до целого.

Решение

Рассмотрим произвольный промежуток времени t . Согласно условиям, теплота Q , которую необходимо отвести от робота за это время, равна $Q = Nt$. Она может

быть определена через удельную теплоту парообразования и плотность воды как

$$Nt = Q = mL = \rho VL,$$

где объем испаренной жидкости V равен произведению расстояния d , которое прошла вода в капилляре на площадь его поперечного сечения:

$$V = dS = vtS.$$

Сопоставляя эти уравнения, получим

$$Nt = \rho vtSL,$$

откуда окончательно

$$v = \frac{N}{\rho SL}.$$

Погрешность 1 мм/с.

Диапазоны

Величина	min	max	Шаг
N , Вт	10	20	1
S , мм ²	0,1	0,2	0,01

Ответ: $v = \frac{N}{\rho SL}$. С учетом порядков $v = \frac{N}{\rho SL} [\cdot 10^6]$.

Задача II.2.5.5. Вагончик (25 баллов)

Темы: закон Ома, кинематика.

Условие

Автоматизированный вагончик движется по двум длинным рельсам, по одному из которых на него подается, а с другого — снимается электрический ток. Каждый метр одного рельса имеет сопротивление r , а двигатели вагончика — полное сопротивление R . Источник питания подает на рельсы напряжение U_0 , а для поддержания нормальной работы двигателей напряжение на них должно быть не ниже U . За какое время, стартовав от источника и двигаясь с постоянной скоростью v , вагончик сможет уехать достаточно далеко, чтобы двигатели перестали работать? Ответ дайте в с, округлив до целых.

Решение

Общее сопротивление цепи, в которую включены двигатели вагончика, равно

$$R_0 = 2r \frac{L}{l} + R,$$

где L — расстояние до источника, $l = 1$ м. Сила тока в цепи $I = U_0/R_0$, а напряжение на двигателе $U_d = IR$ (согласно закону Ома). Подставляя $U_d = U$, $L = vt$, получим:

$$U = U_0 \frac{R}{R_0} = U_0 \frac{R}{2rvt/l + R}.$$

Решая это уравнение относительно t , получим окончательный ответ:

$$t = \frac{R(U_0 - U)}{2rvU} \cdot 1 \text{ м.}$$

Погрешность 1 мин.

Диапазоны

Величина	min	max	Шаг
R , Ом	400	640	40
r , мОм	400	640	40
U , В	200	220	5
U_0 , В	320	360	5
v , м/с	6	8	0,5

Ответ: $t = \frac{R(U_0 - U)}{2rvU} \cdot 1 \text{ м.}$ С учетом порядков $t = \frac{R(U_0 - U)}{2rvU} \cdot [10^3]$.

Задача II.2.5.6. (5 баллов)

Темы: физики России.

Условие

Вектор, описывающий плотность потока энергии в волнах, в англоязычной традиции носит фамилию английского физика, который в 1884 году вывел выражения для данного вектора в случае электромагнитных волн. Десятью годами ранее русский физик и философ вывел аналогичные уравнения для упругих волн, а потому в русской традиции вектор носит и его имя. Помимо выдающихся успехов в теории упругости, он сделал множество значимых открытий в оптике, а также выдвинул гипотезы о качественно правильном характере связи между массой и энергией, позже развитые в знаменитую формулу Эйнштейна $E = mc^2$. Выберите из приведенного списка выдающихся физиков имя этого русского ученого.

1. Владимир Александрович Фок.
2. Николай Алексеевич Умов.
3. Михаил Васильевич Остроградский.
4. Роберт Эмильевич Ленц.
5. Петр Николаевич Лебедев.
6. Александр Александрович Фридман.
7. Анатолий Алексеевич Логунов.
8. Рем Викторович Хохлов.

Ответ: 2.

Третья волна. Задачи 10–11 класса

Задача II.2.6.1. Поглотитель (15 баллов)

Темы: тепловые явления, радиоактивность.

Условие

Быстро движущиеся нейтроны, образующиеся при делении атомных ядер, взаимодействуя с молекулами воды, теряют энергию до значений, соответствующих энергии теплового движения, которая, как правило, много меньше энергии ядерного распада. Оцените, какое число нейтронов, движущихся со скоростью v , должно потерять свою энергию в кювете с водой в виде прямоугольного параллелепипеда со сторонами a, b, d , чтобы вода в этой кювете нагрелась на $\Delta t = 1^\circ\text{C}$. Удельная теплоемкость воды $c = 4,2 \text{ кДж}/(\text{кг}\cdot^\circ\text{C})$, ее плотность $\rho = 1000 \text{ кг}/\text{м}^3$. Масса нейтрона $m = 1,67 \cdot 10^{-27} \text{ кг}$. Ответ дайте в квдрлн (квадриллионах) (10^{15}) штук, округлив до целого.

Решение

Один нейтрон обладает кинетической энергией $K = mv^2/2$. Поскольку эта энергия на несколько порядков величины выше характерной энергии теплового движения, которую можно оценить как $kT \approx 4 \cdot 10^{-21} \text{ Дж}$ для комнатной температуры, можно считать, что вся его энергия передается воде. Для нагрева воды на Δt требуется энергия, равная

$$Q = cm\Delta t = c\rho abd\Delta t.$$

Приравнявая ее nK , получим ответ:

$$n = 2 \frac{c\rho abd\Delta t}{mv^2}.$$

Погрешность 3 квдрлн.

Диапазоны

Величина	min	max	Шаг
v , км/с	10 000	20 000	1000
a , см	40	60	5
b , см	10	16	2
d , см	10	16	2

Ответ: $n = 2 \frac{c\rho abd\Delta t}{mv^2}$. С учетом порядков $n = 2 \frac{c\rho abd\Delta t}{mv^2} \cdot [10^{-24}]$.

Задача II.2.6.2. Дрейф (18 баллов)

Темы: электрический ток, МКТ.

Условие

В некотором полупроводнике концентрация подвижных электронов равна n . Какой должна быть дрейфовая (средняя) скорость электронов в элементе с площадью поперечного сечения S , изготовленном из этого полупроводника, чтобы сила электрического тока в данном сечении была равна I ? Модуль заряда электрона $e = 1,6 \cdot 10^{-19}$ Кл. Ответ дайте в см/с, округлив до целого.

Решение

Рассмотрим произвольный отрезок времени t . При силе тока I за это время через сечение проводника должен пройти заряд $q = It$. Этот заряд равен произведению числа N пересекающих сечение электронов на заряд одного из них e . В то же время при скорости дрейфа v в среднем за время t электроны проходят расстояние $l = vt$. Это значит, что через сечение проходят электроны, содержащиеся в объеме $V = lS = vtS$. Из определения концентрации следует, что их общее число $N = nV$. Совмещая все эти выражения, получим

$$It = q = enV = envtS,$$

откуда легко выражается ответ:

$$v = \frac{I}{Sne}.$$

Погрешность 2 см/с.

Диапазоны

Величина	min	max	Шаг
$n, 10^{18} \text{ м}^{-3}$	6	9	1
$S, \text{ мм}^2$	2, 4	3	0, 1
$I, \text{ мкА}$	1, 5	2, 5	0, 1

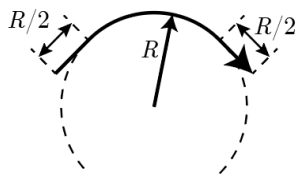
Ответ: $v = \frac{I}{Sne}$. С учетом порядков $v = \frac{I}{Sne} [-10^{-16}]$.

Задача II.2.6.3. Выраз (20 баллов)

Темы: кинематика.

Условие

Квадрокоптер может двигаться по любой траектории с условием, что его скорость ни в одной точке этой траектории не превышает v , а его ускорение не превышает a (при этом направления скорости и ускорения не имеют значения). За какое минимальное время он сможет пройти по траектории, изображенной на рисунке, состоящей из двух прямолинейных участков и четверти окружности радиуса R , если в начальной и в конечной точках коптер должен иметь строго нулевую скорость? Ответ дайте в с, округлив до десятых.



Решение

Общая длина траектории $(4 + \pi/2)R < 6R$. Однако, даже двигаясь все время с набором скорости на расстоянии $6R$ с ускорением a , строго сонаправленным со скоростью, коптер успел бы разогнаться только до

$$v_1 = \sqrt{3aR}.$$

Поскольку эта скорость меньше v , ограничения на максимальную скорость в задаче на самом деле несущественны. В то же время ограничения на скорость накладывает центростремительное ускорение, с которым коптер проходит изогнутый участок: поскольку оно не должно превышать a , скорость на повороте не может быть выше $v_1 = \sqrt{aR}$. Но из формулы

$$\frac{v_1^2 - v_0^2}{2} = al = \frac{aR}{2}$$

можно видеть, что v_1 в точности совпадает со скоростью, которую успевает набрать квадрокоптер на первом прямолинейном и сбросить на втором прямолинейном участках. Таким образом, он должен двигаться равноускоренно на двух участках длиной $R/2$, каждый из которых отнимет время

$$t_1 = \frac{v_1}{a} = \sqrt{\frac{R}{a}}$$

и равномерно на повороте, на что уйдет время

$$t_2 = \frac{\pi R}{2v_1} = \frac{\pi R}{2\sqrt{aR}} = \frac{\pi}{2} \sqrt{\frac{R}{a}}$$

В общей сложности, подобный полет займет время $t_2 + 2t_1$:

$$t = \sqrt{\frac{R}{a}} \left(2 + \frac{\pi}{2} \right).$$

Погрешность 0,2 с.

Диапазоны

Величина	min	max	Шаг
v , м/с	20	40	2
a , м/с ²	3	4	0,2
R , м	15	20	1

Ответ: $t = \sqrt{\frac{R}{a}} \left(2 + \frac{\pi}{2} \right)$.

Задача II.2.6.4. Звездная величина (20 баллов)

Темы: фотометрия, геометрическая оптика.

Условие

Для получения качественного изображения тусклых звезд важно собрать в телескоп максимально возможную долю испущенного такой звездой излучения, что требует больших размеров основного зеркала или линзы телескопа. Яркость звезды в астрономии принято измерять в единицах видимой звездной величины (ВЗВ), увеличение ВЗВ на 1 означает уменьшение световой энергии, испускаемой звездой, в $k = \sqrt[5]{100}$ раз. При помощи телескопа-рефрактора с площадью главного зеркала s получено качественное изображение некоторой звезды. Найдите площадь S главного зеркала второго телескопа, позволяющего получить в таком же качестве изображение звезды, ВЗВ которой больше на n единиц. Ответ дайте в см^2 , округлив до целого.

Решение

Полная мощность светового потока, попадающего в объектив телескопа, определяется как произведение площади S этого объектива на плотность этого потока I . Увеличение видимой звездной величины на n единиц означает уменьшение плотности светового потока в k^n раз. Следовательно, для компенсации этого изменения площадь объектива должна быть в k^n раз увеличена:

$$\frac{S}{s} = k^n.$$

Отсюда

$$S = sk^n = 100^{n/5}s.$$

Погрешность 5 см^2 .

Диапазоны

Величина	min	max	Шаг
$s, \text{ см}^2$	20	50	10
n	2	5	1

Ответ: $S = sk^n = 100^{n/5}s$.

Задача II.2.6.5. Большая линза (22 баллов)

Темы: фотометрия, масса и плотность.

Условие

Яркость звезды принято измерять в единицах видимой звездной величины (ВЗВ), одна единица которой означает изменение световой энергии, испускаемой звездой,

в $k = \sqrt[3]{100}$ раз. Определите, какой массы линзу объектива телескопа-рефрактора пришлось бы использовать при сохранении пропорций и материала линзы, чтобы получить изображение некоторой тусклой звезды в таком же качестве, как было получено изображение на n единиц ВЗВ более яркой звезды при помощи телескопа с объективом, масса которого составляла m . Ответ дайте в т, округлив до десятых.

Решение

Как и в предыдущей задаче, площадь объектива должна быть в k^n раз увеличена. Это требует изменения радиуса объектива в $\sqrt{k^n} = k^{n/2}$ раз. Но при сохранении пропорций объем любого тела и, следовательно, масса такого объектива должны измениться пропорционально кубу радиуса:

$$M = mk^{3n/2}.$$

Погрешность 0, 1 т.

Диапазоны

Величина	min	max	Шаг
m , г	100	200	20
n	7	9	1

Ответ: $vM = k^{3n/2}m = 100^{0,3n}m$. С учетом порядков $M = 100^{0,3n}m \cdot 10^{-6}$.

Задача II.2.6.6. (5 баллов)

Темы: физики России.

Условие

Вектор, описывающий плотность потока энергии в волнах, в англоязычной традиции носит фамилию английского физика, который в 1884 году вывел выражения для данного вектора в случае электромагнитных волн. Десятью годами ранее русский физик и философ вывел аналогичные уравнения для упругих волн, а потому в русской традиции вектор носит и его имя. Помимо выдающихся успехов в теории упругости, он сделал множество значимых открытий в оптике, а также выдвинул гипотезы о качественно правильном характере связи между массой и энергией, позже развитые в знаменитую формулу Эйнштейна $E = mc^2$. Выберите из приведенного списка выдающихся физиков имя этого русского ученого.

1. Владимир Александрович Фок.
2. Александр Александрович Фридман.
3. Рем Викторович Хохлов.
4. Николай Алексеевич Умов.
5. Михаил Васильевич Остроградский.
6. Роберт Эмильевич Ленц.

7. Петр Николаевич Лебедев.
8. Анатолий Алексеевич Логунов.

Ответ: 4.

Инженерный тур

1. Решая задачи инженерного тура первого этапа, участники подтверждают или получают необходимые знания и опыт в области основных понятий беспилотных авиационных систем, физики (в том числе динамики) полета, аэродинамики, математики, что позволяет участникам подготовиться наилучшим образом к решению задач второго этапа, а впоследствии — финальных задач.
2. Решать задачи инженерного тура первого этапа может любой участник самостоятельно.
3. Для решения задач инженерного тура первого этапа требуется знание следующих тем: математика (решение уравнений), физика (кинематика, динамика полета, аэродинамика), основные понятия беспилотных авиационных систем, программирование (Python, C++).

Задача II.3.1. Доставка груза с помощью беспилотного летательного аппарата (10 баллов)

Темы: физика, кинематика.

Условие

Беспилотный летательный аппарат (БЛА) самолетного типа совершает горизонтальный полет на высоте 122 м со скоростью 72 км/ч. На борту БЛА находится полезный груз массой 1 кг. Через некоторое время открываются створки грузогрузового отсека БЛА и груз начинает падение (начальная вертикальная скорость равна нулю).

Определите модуль мгновенной скорости груза при достижении им поверхности земли при условии отсутствия парашюта (сопротивлением воздуха пренебречь, ускорение свободного падения принять равным $9,8 \text{ м/с}^2$).

Ответ округлите до целых чисел.

Решение

Горизонтальная скорость падения груза в начальный момент времени будет равна скорости БЛА — 20 м/с. Так как сопротивление воздуха не учитывается, горизонтальная скорость груза изменяться не будет. Вертикальная скорость в начальный момент времени равна нулю, а в конечный момент времени рассчитывается по формуле равноускоренного движения.

Вертикальное перемещение груза составило 122 м при нулевой начальной скорости:

$$122 = \frac{9,8 \cdot t^2}{2}.$$

Таким образом, время падения груза составило 4,99 с. Зная ускорение свободного

падения, найдем скорость:

$$v_z = g \cdot t = 48,9.$$

Суммарная скорость определяется по теореме Пифагора:

$$V = \sqrt{20^2 + 48,9^2} = 52,83.$$

Ответ: 53.

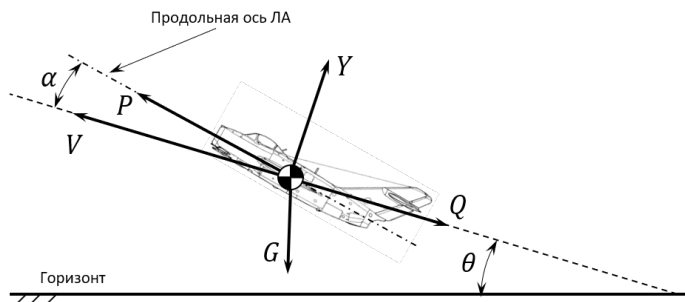
Задача II.3.2. Основы аэродинамики БЛА (20 баллов)

Темы: динамика полета, уравнения, аэродинамика.

Условие

Беспилотный летательный аппарат (БЛА) массой 3,2 кг совершает прямолинейный полет с постоянной скоростью V по траектории, наклоненной под углом θ к горизонту.

Схема сил, действующих на БЛА, обозначена на рисунке ниже.



Сила лобового сопротивления $Q = c_x(\alpha) \frac{\rho V^2}{2} S$ направлена противоположно скорости ЛА, подъемная сила — перпендикулярно скорости ЛА, а тяга P — по продольной оси ЛА.

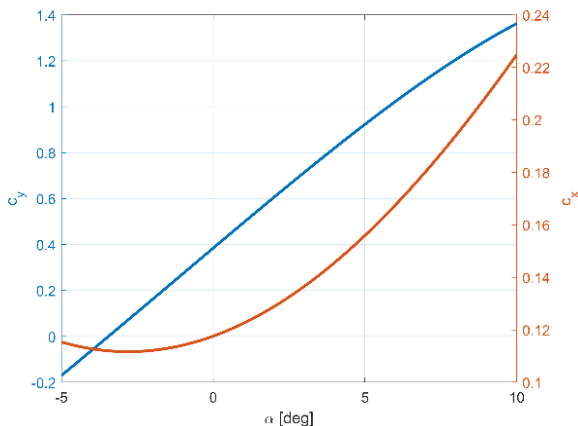
Плотность воздуха $\rho = 1,225$ кг/м³, характерная площадь крыла $S = 0,36$ м².

Поляра ЛА аппроксимирована на интервале $\alpha \in [-5, 10]^\circ$ следующими полиномами:

$$\begin{aligned} c_x(\alpha) &= c_x^{(2)}(\alpha)^2 + c_x^{(1)}(\alpha) + c_x^{(0)}, \\ c_y(\alpha) &= c_y^{(2)}(\alpha)^2 + c_y^{(1)}(\alpha) + c_y^{(0)}, \end{aligned}$$

где коэффициенты $c_x^{(n)}$ и $c_y^{(n)}$ одинаковы для всех вариантов для избежания ошибок при задании параметров. Коэффициенты подразумевают задание угла атаки в градусах.

Зависимость значений этих коэффициентов от угла атаки называется поляррой БЛА, и в графическом представлении она имеет вид.



Определите значение угла атаки α ($^\circ$) для заданной поляры БЛА и условий полета (перечислены ниже), которые потребуются для установившегося полета (силы уравновешены, ускорения равны нулю).

- $V = 35$ км/ч;
- $\theta = 20^\circ$;
- коэффициенты $c_x^{(2)}, \dots, c_x^{(0)} = 0,0007 \ 0,0041 \ 0,1176$;
- коэффициенты $c_y^{(2)}, \dots, c_y^{(0)} = -0,0002 \ 0,1113 \ 0,3847$;
- ускорение свободного падения $g = 9,81$ м/с²

Обратите внимание, что ответ нужно указать в градусах с точностью до второго знака после запятой.

Решение

Запишем уравнение баланса сил в проекции на вертикальную ось связанной системы координат:

$$Y \cdot \cos(\alpha) + Q \cdot \sin(\alpha) - G \cdot \cos(\theta) = 0.$$

В полученном уравнении Y и Q зависят от угла атаки α .

Таким образом, в этом уравнении только одна неизвестная величина — угол атаки.

Составим программу для решения этого уравнения.

Пример программы-решения

Ниже представлено решение на языке C++.

```

1 #include <iostream>
2 #include <string>
3 #include <cmath>
4 #define PI 3.1415926

```

```

5  using namespace std;
6
7  double c(double cxdn[3],double a)
8  {
9      double cn = 0;
10     for (int i=0;i<3;i++){
11         cn=cn+cdn[i]*pow(a,i);
12     };
13     return cn;
14 }
15
16 double get_alpha(double G,double k,double f,double cxdn[3],double cydn[3])
17 {
18     double fi=f*(PI/180.0);
19     double min_error = 9999999999999999;
20     double ma = -5;
21     for (double i=-5.0;i<10.0;i=i+0.001){
22         double ang=i*(PI/180.0);
23         //double kkk = tan(ang) - ( G*cos(fi) - c(cydn,i)*k ) / ( c(cxdn,i)*k
24         ↪ +G*sin(fi) );
25         double kkk = c(cydn,i)*k * cos(ang) + c(cxdn,i)*k * sin(ang) - G*cos(fi +
26         ↪ ang);
27         if ( abs(kkk) < min_error){
28             min_error = abs(kkk);
29             ma = i;
30         }
31     }
32     return ma;
33 }
34
35 int main()
36 {
37     double cxdn[3];
38     double cydn[3];
39     double cy2,cy1,cy0;
40     double V, f;
41     const double g=9.81;
42     const double p=1.225;
43     const double S=0.36;
44     const double m=3.2;
45     cxdn[0] = 0.1176;
46     cxdn[1] = 0.0041;
47     cxdn[2] = 0.0007;
48
49     cydn[0] = 0.3847;
50     cydn[1] = 0.1113;
51     cydn[2] = -0.0002;
52     V = 35/3.6;
53     double k=((p * V * V)/2)*S;
54     double G=m*g;
55     double a = get_alpha(G,k,f,cxdn,cydn);
56     double cx=c(cxdn,a);
57     double cy=c(cydn,a);
58     double O=(f*PI)/180.0;
59     double A = (a*PI)/180.0;
60     double P=G*sin(O + A) + cx*k*cos(A) - cy*k*sin(A);
61     printf("%.5f",5,a);
62     cout << " ";
63     printf("%.5f",5,P);
64     return 0;

```

63 }

Ответ: 8,44.

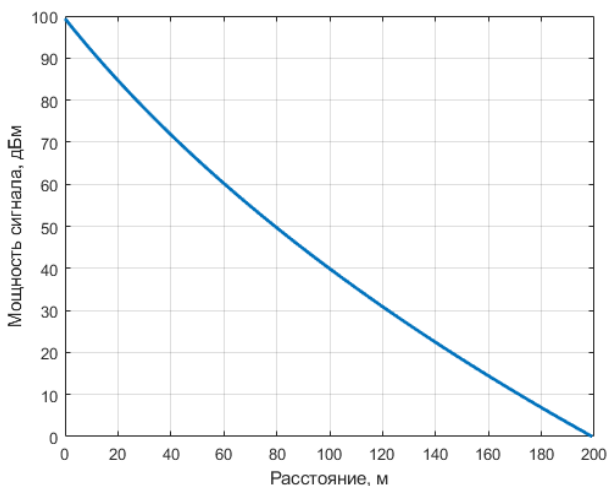
Задача П.3.3. Определение местоположения методом трилатерации (25 баллов)

Темы: математика, уравнения, координаты.

Условие

В процессе полета БЛА получает информационный сигнал от трех вышек связи и измеряет мощность этого сигнала. Известна экспериментальная зависимость расстояния до источника (вышки связи) от мощности полученного сигнала.

$$distance = -power^3 \cdot 1,1 \cdot 10^{-1} + power^2 \cdot 0,0096 - power \cdot 2,85 + 199,19.$$



Напишите программу на языке C++ или Python, определяющую местоположение БЛА по известным координатам вышек связи в трехмерном декартовом пространстве и по заданным измерениям мощности сигнала до каждой из вышек.

Гарантируется, что расстояние от БЛА до вышек связи не превышает 250 м, и что задача имеет решение. Кривизну поверхности Земли при расчетах не учитывайте.

Формат входных данных

Массив координат (x, y) вышек связи в м и измеренной мощности (P) сигнала (дБм), заданных последовательно, через пробелы. Гарантируется, что все значения координат и мощностей — положительные и целые величины (без дробной части).

XXX YYY PPP XXX YYY PPP XXX YYY PPP

Формат выходных данных

Значения двух координат (x, y) БЛА в декартовом пространстве (в м) с точностью до второго знака после запятой, записанные через пробел. Если задача имеет несколько решений, правильным ответом считается среднее арифметическое всех решений.

Запись значений должна соответствовать формату: три целых числа, точка и сотые доли значения (например, значение 2,1 записывается как 002.10).

XXX.XX YYY.YY

Примеры

Пример №1

Стандартный ввод
085 072 050 197 116 001 127 095 029
Стандартный вывод
011.95 055.88

Решение

Словесное описание алгоритма решения:

1. считать входные данные и пересчитать мощность сигнала в дальность до вышек;
2. составить уравнения трех окружностей — получится система из трех уравнений с двумя неизвестными;
3. сложить первые два уравнения системы;
4. вычесть второе уравнение из третьего;
5. выразить одну из координат точки пересечения (получится квадратное уравнение);
6. решить квадратное уравнение и найти парные координаты;
7. определить, какая из двух точек является пересечением всех трех окружностей.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```

1 import numpy
2 from numpy import sqrt, dot, cross
3 from numpy.linalg import norm
4
5
6 tower1 = [104, 156, 0, 21]
7 tower2 = [237, 157, 0, 6]
8 tower3 = [190, 200, 0, 1]
9 cfs = [199.19, -2.85, 0.0096, -1.1e-5]
10
```

```

11 r1 = cfs[0] * 1 + cfs[1] * tower1[3] + cfs[2] * tower1[3] ** 2 + cfs[3] *
   ↪ tower1[3] ** 3
12
13 r2 = cfs[0] * 1 + cfs[1] * tower2[3] + cfs[2] * tower2[3] ** 2 + cfs[3] *
   ↪ tower2[3] ** 3
14
15 r3 = cfs[0] * 1 + cfs[1] * tower3[3] + cfs[2] * tower3[3] ** 2 + cfs[3] *
   ↪ tower3[3] ** 3
16
17 P1 = tower1[:3]
18 P2 = tower2[:3]
19 P3 = tower3[:3]
20
21 def trilaterate(P1, P2, P3, r1, r2, r3):
22     temp1 = []
23     for i in range(len(P1)):
24         temp1.append(P2[i] - P1[i])
25     e_x = temp1 / norm(temp1)
26     temp1_abs = []
27     temp2 = []
28     for i in range(len(P1)):
29         temp2.append(P3[i] - P1[i])
30     i = dot(e_x, temp2)
31     temp3 = temp2 - i * e_x
32     print(f'temp3: {temp3}')
33     e_y = temp3 / norm(temp3)
34     e_z = cross(e_x, e_y)
35     # d = norm(P2-P1)
36     d = norm(temp1)
37     j = dot(e_y, temp2)
38     x = (r1 * r1 - r2 * r2 + d * d) / (2 * d)
39     y = (r1 * r1 - r3 * r3 - 2 * i * x + i * i + j * j) / (2 * j)
40     temp4 = r1 * r1 - x * x - y * y
41     print(f'temp4: {temp4}')
42     z = sqrt(temp4)
43     p_12_a = P1 + x * e_x + y * e_y + z * e_z
44     p_12_b = P1 + x * e_x + y * e_y - z * e_z
45     return p_12_a, p_12_b
46
47 a, b = trilaterate(P1, P2, P3, r1, r2, r3)
48 x = round((a[0] + b[0]) / 2, 2)
49 y = round((a[1] + b[1]) / 2, 2)
50 z = round((a[2] + b[2]) / 2, 2)
51 print(str(x).zfill(6), str(y).zfill(6), str(z).zfill(6))

```

Задача II.3.4. Определение скорости азимутального разворота БЛА (25 баллов)

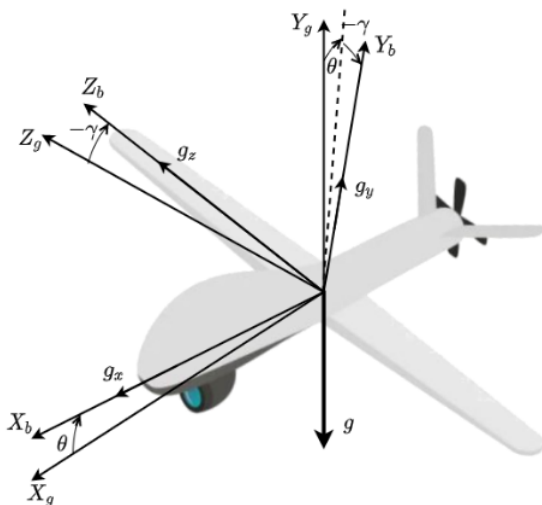
Темы: физика полета, уравнения.

Условие

Определение углов ориентации БЛА в процессе полета является нетривиальной задачей. Одним из источников информации об углах ориентации БЛА является датчик ускорений — акселерометр. Этот датчик позволяет измерять проекции полного вектора ускорения БЛА на его связанные оси: продольную, поперечную и вертикальную. Таким образом, зная расположение вектора полного ускорения БЛА, можно

определить углы его ориентации.

Предположим, что БЛА самолетного типа совершает азимутальный разворот с постоянной скоростью. При этом известны проекции вектора ускорения свободного падения на оси связанной системы координат.



На рисунке обозначены: $X_g Y_g Z_g$ — оси нормальной земной системы координат; $X_b Y_b Z_b$ — оси связанной системы координат БЛА; g — вектор ускорения свободного падения; $g_x g_y g_z$ — проекции ускорения свободного падения на оси $X_b Y_b Z_b$; θ , γ — углы тангажа и крена соответственно.

Значение мгновенного радиуса разворота БЛА на постоянной высоте можно рассчитать по формуле:

$$R = \frac{V^2}{g \cdot \tan(\gamma)},$$

где V — истинная скорость БЛА; γ — угол крена; g — ускорение свободного падения.

Определите угловую скорость азимутального разворота БЛА ($^{\circ}/c$) по заданным значениям истинной скорости полета ($15,1$ м/с) и проекциям вектора ускорения свободного падения на его продольную ($-1,15$ м/с²), поперечную ($1,23$ м/с²) и вертикальную оси ($-9,71$ м/с²).

Ответ укажите с точностью до второго знака после запятой.

Решение

Словесное описание алгоритма решения:

1. определить модуль ускорения свободного падения (по условию задачи он отличается от $9,81$);
2. при помощи тригонометрии найти угол крена;

3. рассчитать радиус разворота;
4. определить угловую скорость как отношение линейной скорости к радиусу.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```
1 import math
2 arr = [float(i) for i in input().split()]
3 V = arr[0]
4 Fx = arr[1]
5 Fy = arr[2]
6 Fz = arr[3]
7 F = math.sqrt(Fx**2 + Fy**2 + Fz**2)
8 gamma = math.asin(Fz/F)
9 g = 9.81
10 R = V ** 2 / (g * math.tan(gamma))
11 dps = V / R
12 print("{:3.2f}".format(dps*180.0/math.pi))
```

Ответ: 4,68.

Работа наставника НТО на втором отборочном этапе

На втором отборочном этапе участникам предлагаются индивидуальные и командные задачи в рамках выбранных профилей. Для подготовки к нему наставник может использовать следующие рекомендуемые форматы и мероприятия:

- Подготовка по образовательным программам НТО по ряду технологических направлений.
- Разбор задач второго отборочного этапа НТО прошлых лет.
- Прохождение онлайн-курсов по разбору задач НТО прошлых лет.
- Прохождение онлайн-курсов, рекомендованных разработчиками профилей.
- Разбор материалов для подготовки к профилям.
- Практикумы. Для организации практикумов возможно использовать разные подходы или их комбинации:
 - Проведение практикумов по описаниям на страницах профилей и материалов для подготовки.
 - Декомпозиция задач заключительных этапов прошлых лет для выделения наиболее актуальных элементов и их изучения.
 - Анализ технических знаний и навыков (hard skills), требуемых для конкретного профиля, и самостоятельная разработка или поиск занятия для развития наиболее актуальных из них.
 - Посещение практикумов на площадках подготовки и онлайн-мероприятий от разработчиков профилей. Объявления о таких мероприятиях публикуются в группах НТО в VK и в телеграм-канале для наставников НТО (https://t.me/kruzhok_association).

Второй отборочный этап

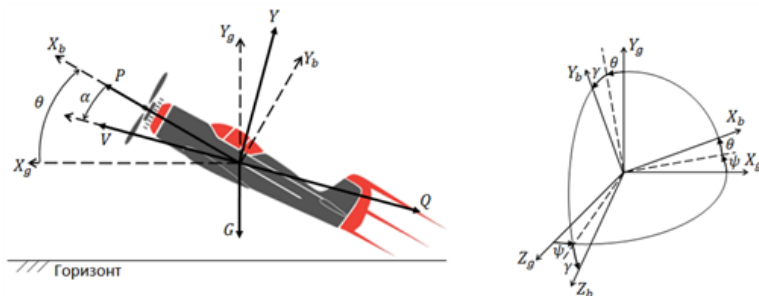
Задача IV.1. Определение высоты полета БЛА (20 баллов)

Темы: динамика полета, физика, решение уравнений.

Условие

Динамика полета БЛА самолетного типа в значительной мере зависит от его аэродинамических характеристик. При проектировании БЛА методом экспериментальных продувок фюзеляжа БЛА и его крыльев определяют поляры – зависимости аэродинамических коэффициентов подъемной силы и силы лобового сопротивления от угла атаки. В то же время силы лобового сопротивления и подъемная будут зависеть не только от аэродинамических коэффициентов, но и от скорости полета, плотности среды (воздуха) и аэродинамической площади крыла.

Допустим, что БЛА совершает полет с постоянной скоростью.



Сила лобового сопротивления Q направлена противоположно вектору путевой скорости V БЛА и равна:

$$Q = cx(\alpha)V^22S,$$

где $cx(\alpha)$ – аэродинамический коэффициент (зависящий от текущего угла атаки);

S – характерная площадь обтекаемой поверхности БЛА;

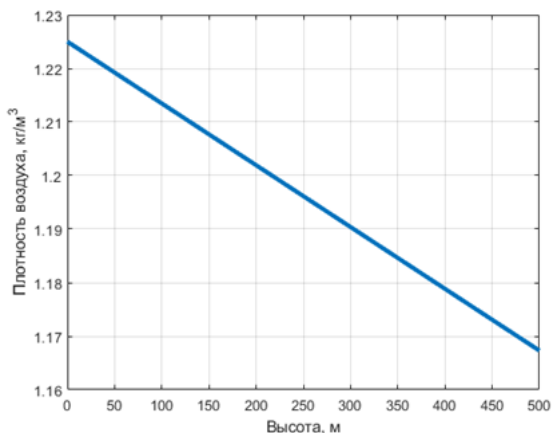
ρ – плотность воздуха.

Подъемная сила Y направлена перпендикулярно вектору V путевой скорости БЛА и равна:

$$Y = cy(\alpha)\frac{\rho V^2}{2}S.$$

Тяга P направлена по продольной оси БЛА, а сила тяжести GG – вертикально вниз. Оси связанной системы координат (СК) $OX_bY_bZ_b$ последовательно повернуты относительно осей географической СК $OX_gY_gZ_g$ (ось OX_g направлена на север, ось OZ_g – на восток) на углы курса ψ , тангажа θ и крена γ . Угол атаки α – угол между вектором путевой скорости V и продольной осью БЛА OX_b .

Зависимость плотности воздуха от высоты на малых интервалах может быть аппроксимирована линейным законом.



Решив уравнения баланса сил, найдите выражение для текущей высоты полета БЛА в зависимости от:

- силы тяжести G ;
- скорости полета V ;
- угла атаки α ;
- характерной площади крыла S ;
- аэродинамических коэффициентов $c_x(\alpha)$, $c_y(\alpha)$;
- плотности воздуха на высоте 0 м ρ_0 ;
- плотности воздуха на высоте 500 м ρ_{500} ;
- угла тангажа θ .

При вводе формулы на платформе степик используйте следующие обозначения:

- силы тяжести G ;
- скорости полета V ;
- угла атаки a ;
- характерной площади крыла s ;
- аэродинамических коэффициентов $c_x(a)$, $c_y(a)$;
- плотности воздуха на высоте 0 м ρ_0 ;
- плотности воздуха на высоте 500 м ρ_{500} ;
- угла тангажа t ;
- косинус, синус угла $\cos(a)$, $\sin(t)$;
- степень $a^2 = a * a$;
- дробь $(a - t) / (V + s)$;
- умножение $a * t$;
- константы 3 .

Порядок слагаемых и множителей не имеет значения. Последовательность математических операций определяется по обычным правилам. Требуется ввести соотношение для получения значения высоты в м.

Платформа Stepik автоматически проверяет формулы на математическое соответствие.

Решение

1. Запишем зависимость плотности воздуха от высоты на малых интервалах и выразим высоту H :

$$\rho = k \cdot H + \rho_0 \Rightarrow H = \frac{\rho - \rho_0}{k}.$$

2. Найдем коэффициент k из условия, что на высоте 500 м плотность воздуха будет равна ρ_{500} :

$$\rho_{500} = k \cdot 500 + \rho_0 \Rightarrow k = \frac{\rho_{500} - \rho_0}{500}.$$

3. Подставим выражение для коэффициента k в формулу для высоты H :

$$H = 500 \cdot \frac{\rho - \rho_0}{\rho_{500} - \rho_0}.$$

4. Найдем выражение для плотности воздуха ρ из уравнения баланса сил:

$$\begin{aligned} Y \cdot \cos \cos(\alpha) + Q \cdot \sin \sin(\alpha) - G \cdot \cos \cos(\theta) &= \\ = 0c_y(\alpha) \frac{\rho V^2}{2} S \cdot \cos(\alpha) + c_x(\alpha) \frac{\rho V^2}{2} S \cdot \sin \sin(\alpha) - G \cdot \cos \cos(\theta) &= \\ = 0 \frac{\rho V^2}{2} S (c_x(\alpha) \cdot \sin \sin(\alpha) + c_y(\alpha) \cdot \cos \cos(\alpha)) - G \cdot \cos \cos(\theta) &= \\ = 0 \frac{\rho V^2}{2} S (c_x(\alpha) \cdot \sin \sin(\alpha) + c_y(\alpha) \cdot \cos \cos(\alpha)) = G \cdot \cos \cos(\theta) \rho &= \\ = \frac{2G \cdot \cos(\theta)}{V^2 S (c_x(\alpha) \cdot \sin \sin(\alpha) + c_y(\alpha) \cdot \cos \cos(\alpha))}. \end{aligned}$$

5. Подставим полученное выражение для плотности воздуха ρ в формулу для высоты H :

$$H = 500 \cdot \frac{\frac{2G \cdot \cos(\theta)}{V^2 S (c_x(\alpha) \cdot \sin \sin(\alpha) + c_y(\alpha) \cdot \cos \cos(\alpha))} - \rho_0}{\rho_{500} - \rho_0}.$$

6. Полученное выражение для высоты H нужно записать согласно обозначениям из условия.

Ответ: $(500 * (2 * G * \cos(t)) / (s * V * V * (c_x(a) * \sin(a) + c_y(a) * \cos(a))) - ro0) / (ro500 - ro0)$.

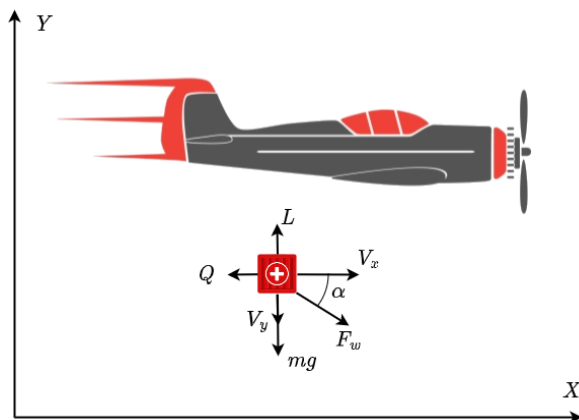
Задача IV.2. Расчет аэродинамики груза (20 баллов)

Темы: динамика полета, физика, численное интегрирование.

Условие

При решении задачи автоматической доставки грузов при помощи БЛА очень важно точно знать аэродинамические параметры груза (и парашюта) для осуществления точной доставки.

Предположим, что БЛА осуществляет сброс груза массой $m = 0,8$ кг на высоте $H = 100$ м. Начальная вертикальная скорость груза $V_y = 0$. Начальная горизонтальная скорость груза равна скорости полета БЛА в момент сброса $V_x = 72$ км/ч.



В процессе падения на груз действуют силы сопротивления воздуха: горизонтальная Q и вертикальная L :

$$Q = 0,5 \cdot c_x \cdot \rho \cdot V_x^2 \cdot S;$$

$$L = 0,5 \cdot c_y \cdot \rho \cdot V_y^2 \cdot S,$$

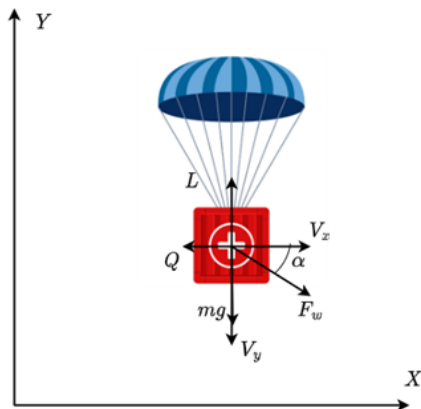
где $c_x = c_y = 1,1$ — аэродинамический коэффициент;

$S = 0,0063$ м² — характерная площадь обтекаемой поверхности груза;

ρ — плотность воздуха.

На груз действует также сила ветра F_w , направленная под углом α к плоскости горизонта.

Через 1,5 с после сброса раскрывается парашют груза.



После раскрытия парашюта аэродинамика груза вдоль вертикальной оси изменится (динамика в плоскости горизонта остается неизменной):

$$L = 0,5 \cdot c_y \cdot \rho \cdot V_y^2 \cdot S_p,$$

где $c_y = 3,4$ — аэродинамический коэффициент;

$S_p = 0,08 \text{ м}^2$ — характерная площадь обтекаемой поверхности груза с парашютом.

Рассчитайте время падения груза в с (считается время с момента сброса до момента контакта с поверхностью Земли) и полную скорость груза в км/ч в момент контакта с Землей.

Рассчитанные величины требуется указать с точностью до второго знака после запятой.

При расчете считать, что:

- раскрытие парашюта происходит мгновенно через 1,5 с после сброса груза;
- угловое движение груза не учитывается;
- высота подстилающей поверхности равна нулю;
- сила ветра постоянна $F_w = 5$;
- направление ветра постоянное $\alpha = 30^\circ$;
- плотность воздуха постоянная и равна $\rho = 1,22 \text{ кг/м}^3$;
- ускорение свободного падения $g = 9,81 \text{ м/с}^2$;
- величина сил сопротивления зависит от квадрата скорости (движение груза не равноускоренное).

Решение задачи предполагает использование численного интегрирования уравнений движения груза.

Максимальный балл за задачу — 20. Рассчитанные величины оцениваются отдельно (10 баллов начисляется за правильный расчет времени падения груза; 10 баллов начисляется за правильный расчет полной скорости груза в момент контакта с Землей).

Решение

Падение груза после его сброса состоит из двух этапов: свободного падения и спуска на парашюте. Поэтому для решения задачи нужно рассмотреть оба этапа.

1. Рассмотрим этап свободного падения.

Воспользуемся вторым законом Ньютона:

$$ma_x = -Q + F_w \cdot \cos \alpha \quad ma_y = L - mg - F_w \cdot \sin(\alpha).$$

Выразим проекции ускорения:

$$a_x = \frac{-Q + F_w \cdot \cos \alpha}{m} \quad a_y = \frac{L - mg - F_w \cdot \sin(\alpha)}{m}.$$

Силы Q и L в выражениях для ускорений будут иметь вид:

$$Q = 0,5 \cdot c_x \cdot \rho \cdot V_x^2 \cdot S,$$

$$L = 0,5 \cdot c_y \cdot \rho \cdot V_y^2 \cdot S.$$

Так как по условию движение груза не равноускоренное, и необходимо использовать численное интегрирование, запишем выражения для координат x и y и скоростей V_x и V_y груза сразу с учетом этого:

$$\begin{aligned} x(i) &= x(i-1) + v_x(i-1) \cdot \Delta t \\ y(i) &= y(i-1) + v_y(i-1) \cdot \Delta t \\ v_x(i) &= v_x(i-1) + a_x(i-1) \cdot \Delta t \\ v_y(i) &= v_y(i-1) + a_y(i-1) \cdot \Delta t. \end{aligned}$$

Соответственно в выражениях для сил Q и L проекции скорости на оси x и y также будут обновляемыми в процессе численного интегрирования

$$Q = 0,5 \cdot c_x \cdot \rho \cdot V_x^2(i-1) \cdot S,$$

$$L = 0,5 \cdot c_y \cdot \rho \cdot V_y^2(i-1) \cdot S.$$

Поскольку по условию свободное падение длится 1,5 с, нужно задать количество точек интегрирования:

$$N = \frac{1,5}{\Delta t}.$$

Значения, полученные в последней точке, по результатам интегрирования станут стартовыми для второго этапа падения груза.

2. Рассмотрим этап спуска на парашюте.

Все формулы выглядят так же, как и на первом этапе, за исключением формулы для L :

$$L = 0,5 \cdot c_y \cdot \rho \cdot V_y^2(i-1) \cdot S_p.$$

Интегрирование на втором этапе следует проводить до тех пор, пока высота груза не станет равной 0.

В результате будут получены время падения груза и проекции скорости. Полную скорость можно рассчитать по формуле:

$$V = \sqrt{V_x^2 + V_y^2}.$$

Таким образом можно составить программу, которая будет выполнять расчет времени падения груза и скорости груза в момент контакта с поверхностью.

Пример программы-решения

```

1 function [time, x, y, vx, vy] = cargo_calc(dt)
2     y = 100;
3     x = 0;
4     time = 0;
5     vx = 72 / 3.6;
6     vy = 0;
7
8     cx = 1.1;
9     cy = 3.4;
10    S = 0.0063;
11    Sp = 0.08;
12    Fw = 5;
13    m = 0.8;
14    g = 9.81;
15    a = 30; % deg
16    ro = 1.22;
17
18    % free fall
19    Nsim = 1.5 / dt;
20    for i = 2:Nsim
21        q = cx * ro * vx(i-1)^2 * S / 2;
22        hor = -q + Fw * cosd(a);
23        l = cx * ro * vy(i-1)^2 * S / 2;
24        ver = l - m * g - Fw * sind(a);
25        x(i) = x(i-1) + vx(i-1) * dt;
26        y(i) = y(i-1) + vy(i-1) * dt;
27        vx(i) = vx(i-1) + hor / m * dt;
28        vy(i) = vy(i-1) + ver / m * dt;
29
30        time(i) = i*dt;
31    end
32    h = y(i);
33    while h >= 0
34        i = i + 1;
35        q = cx * ro * vx(i-1)^2 * S / 2;
36        hor = -q + Fw * cosd(a);
37        l = cy * ro * vy(i-1)^2 * Sp / 2;
38        ver = l - m * g - Fw * sind(a);
39        x(i) = x(i-1) + vx(i-1) * dt;
40        y(i) = y(i-1) + vy(i-1) * dt;
41        vx(i) = vx(i-1) + hor / m * dt;
42
43        vy(i) = vy(i-1) + ver / m * dt;
44
45        h = y(i);
46        time(i) = i*dt;
47    end
48    fprintf("Time elapsed: %f\n", time(end))
49    fprintf("Velocity: %f %f\n", vx(end), vy(end))
50    fprintf("Total velocity: %f\n", sqrt(vx(end)^2 + vy(end)^2) * 3.6)
51 end

```

Поскольку по условию ответ округляется до второго знака после запятой, то есть до сотых долей, шаг интегрирования Δt (dt — в программе) следует брать еще меньше, например, 0,0001.

Ответ: время падения груза: 12,05 с; скорость груза в момент контакта: 117,80 км/ч.

Задача IV.3. Построение оптимального поискового маршрута БЛА (30 баллов)

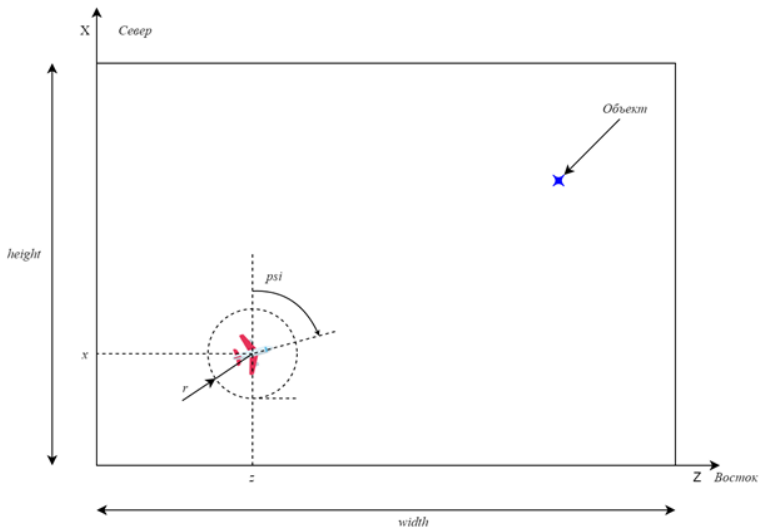
Темы: моделирование полета, программирование (Python), оптимизация.

Условие

При решении большого спектра задач при помощи БЛА необходимо выполнять поиск объектов на поверхности. Для этого необходим алгоритм, который позволит проложить оптимальный маршрут поиска этого объекта на поверхности.

Данные

Перед началом полета на борт БЛА загружается информация об области поиска объекта в виде размеров `height` и `width`. Предположим, что БЛА оснащен системой технического зрения, позволяющей определить наличие объекта в зоне подстилающей поверхности радиусом r непосредственно под БЛА. Система автоматического управления БЛА принимает команды управления с частотой 10 Гц в виде линейной скорости V и угла крена $roll$. А система навигации позволяет определить текущий курс БЛА и его координаты относительно области поиска — psi , x , z .



Задача

Разработайте класс `solver`, в котором должен быть метод `solve`, позволяющий на основе полученных от навигационной системы данных рассчитать необходимые

угол крена (`roll`), линейную скорость (`V`) и логическое значение сообщающее нужно ли на данном шаге проводить сканирование подстилающей поверхности.

Конструктор класса `solver` должен принимать размеры поля в виде объекта `tuple` с двумя значениями шириной и высотой.

Пример: `solver_ex = solver(500, 500)`.

Метод `solve` должен принимать параметр `telemetry`, представляющий собой словарь со следующими полями:

1. `telemetry["X"]` — текущее значение координаты `x` БЛА;
2. `telemetry["Z"]` — текущее значение координаты `z` БЛА;
3. `telemetry["Heading"]` — текущее значение угла `psi` БЛА.

Возвращать метод `solve` должен три значения: скорость, угол крена и значение означающее необходимость сканирования (`True` означает что необходимо произвести сканирование).

```
return float(v), float(roll), bool(scan)
```

На основе выработанных методом `solve` команд будет проводиться симуляция осмотра области поиска объекта до обнаружения объекта либо до истечения лимита по времени симуляции.

Ограничения

Так как поиск выполняется с помощью БЛА самолетного типа, есть ограничения на его линейную скорость и максимальный угол крена.

Скорость должна находиться в диапазоне $15 \text{ м/с} \leq V \leq 25 \text{ м/с}$ для того, чтобы БЛА мог поддерживать постоянную высоту полета.

Угол крена должен находиться в диапазоне $-20^\circ \leq roll \leq 20^\circ$, чтобы избежать сваливания БЛА.

Частота, с которой проводится симуляция, составляет 10 Гц, команды отрабатываются мгновенно, т. е. сразу после выработки новых значения скорости и крена следующий шаг симуляции будет выполнен с этими значениями.

Радиус сканирования $r = 30 \text{ м}$, и его величина не меняется. Вылет за пределы области поиска допустим. Начальное положение БЛА во всех тестах — $(30, 30)$. Начальный угол курса равен нулю.

Критерии оценивания

Для оценки эффективности алгоритма используется параметр эффективность сканирования, который рассчитывается по следующей формуле:

$$es = \frac{S_s}{N_s},$$

где es — эффективность сканирования;

S_s — площадь просканированной области поиска;

S — площадь области поиска;

N_s — количество сканирований.

Эффективность сканирования характеризует степень эффективности поиска. Повторное сканирование областей, так же как и сканирование за пределами области поиска, уменьшают значение критерия.

Эффективность сканирования позволяет оценить, какую долю области сканирования вы в среднем проверяли при каждом сканировании. Эта величина падает, если вы сканируете уже просканированные участки или участки, выходящие за пределы области сканирования (вылетать за пределы области сканирования допустимо).

Данный критерий рассматривается только в том случае, если объект был найден за 200 с симуляции (один шаг симуляции равен 0,1 с).

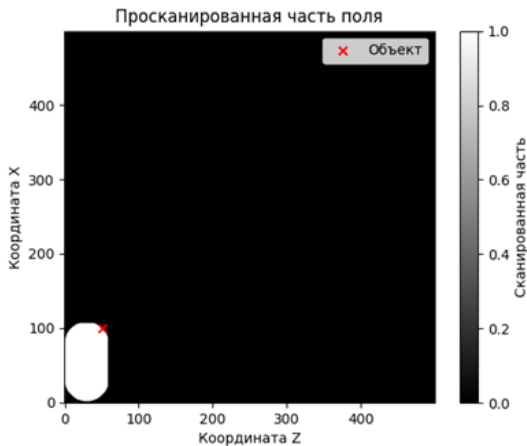
Максимальный балл за задачу — 30. Такой балл можно получить за прохождение всех тестов со средним результатом эффективности сканирования 1,1, далее за каждое уменьшение эффективности на 0,02 единицы снимается по одному баллу. Оценивается осредненный результат по всем тестам.

В расчет берется последнее решение, а округление эффективности сканирование происходит в пользу участников, например, значение 1,081 будет округляться до 1,1 и даст участникам максимальный балл.

Моделирование на GitHub

Для проверки своего решения оффлайн вы можете воспользоваться кодом репозитория на GitHub: https://github.com/Kortium/search_object_tast.

Он позволяет визуализировать ваше решение в виде небольшой диаграммы, в которой черным обозначается необследованная часть зоны поиска, а белым — обследованная. Искомый объект обозначается красным крестом.



Формат входных данных

Размер области поиска по осям X , Z и координаты объекта, заданные через пробел.

maxX maxZ objectX objectZ

Формат выходных данных

По завершении выполнения программы автоматически выводится текст следующего формата

```
{'Time Spent': 30.000000000000156, 'Scanned Percentage': 16.3512,
'Scan counter': 300, 'Scan effectiveness': 0.054504}
```

и логическое значение — был ли найден объект (True, если объект был обнаружен). True

После отправки решения вы увидите аналогичный вывод для каждого из тестов, что поможет вам оценить свое решение.

Тесты

https://disk.yandex.ru/d/VOMgPj-eLv00Fw/Тесты_задача_3.docx

Решение

Словесное описание алгоритма решения.

1. Задать необходимые переменные и константы с их начальными значениями в конструкторе класса `solver`.
2. Сформировать траекторию полета БЛА для реализации эффективного поиска объекта (траектория должна покрывать максимальную площадь поиска с учетом ограничения по времени).
3. Разработать алгоритм переключения между фазами полета БЛА для реализации траектории.
4. Рассчитать требуемую частоту сканирования с учетом ограничения на эффективность сканирования.
5. Проверить решение на тестовых примерах.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```
1 class solver:
2     EDGE = 80
3     DELAY = 45
4     SCAN_PERIOD_METERS = 60
5
6     def __init__(self, field_size):
7         self.x_direction = 1
8         self.z_direction = 1
9         self.horizontal_scan = True
```

```

10     self.prev_x = 0
11     self.prev_z = 0
12     self.delay = 0
13     self.field_size = field_size
14     self.range_since_last_scan = self.SCAN_PERIOD_METERS
15
16     def solve(self, telemetry):
17         current_x_speed = telemetry["X"] - self.prev_x
18         current_z_speed = telemetry["Z"] - self.prev_z
19         self.range_since_last_scan += sqrt(current_x_speed**2 +
20 ↪ current_z_speed**2)
21         scan = False
22         if self.range_since_last_scan > self.SCAN_PERIOD_METERS:
23             scan = True
24             self.range_since_last_scan = 0
25         if self.delay > 0:
26             self.delay -= 1
27             return 25, 0, scan
28         self.prev_x = telemetry["X"]
29         self.prev_z = telemetry["Z"]
30         v=25
31         r=0
32         if self.horizontal_scan:
33             if telemetry["Z"] > self.field_size[0] - self.EDGE and
34 ↪ self.z_direction > 0:
35                 self.horizontal_scan = False
36                 self.delay = self.DELAY
37             if telemetry["Z"] < self.EDGE and self.z_direction < 0:
38                 self.horizontal_scan = False
39                 self.delay = self.DELAY
40             if self.x_direction > 0:
41                 if current_x_speed < 0:
42                     v=15
43                     r=-20
44                 else:
45                     if current_z_speed > 0:
46                         v=15
47                         r=-20
48                     if current_z_speed < 0:
49                         v=15
50                         r=20
51             else:
52                 if current_x_speed > 0:
53                     v=15
54                     r=20
55                 else:
56                     if current_z_speed > 0:
57                         v=15
58                         r=20
59                     if current_z_speed < 0:
60                         v=15
61                         r=-20
62             else:
63                 if telemetry["X"] > self.field_size[1] - self.EDGE and
64 ↪ self.x_direction > 0:
65                     self.horizontal_scan = True
66                     self.delay = self.DELAY
67                 if telemetry["X"] < self.EDGE and self.x_direction < 0:
68                     self.horizontal_scan = True
69                     self.delay = self.DELAY

```

```

67     if self.z_direction > 0:
68         if current_z_speed < 0:
69             v=15
70             r=20
71         else:
72             if current_x_speed > 0:
73                 v=15
74                 r=20
75             if current_x_speed < 0:
76                 v=15
77                 r=-20
78     else:
79         if current_z_speed > 0:
80             v=15
81             r=-20
82         else:
83             if current_x_speed > 0:
84                 v=15
85                 r=-20
86             if current_x_speed < 0:
87                 v=15
88                 r=20
89     if self.x_direction > 0 and telemetry["X"] > self.field_size[1] -
90     ↪ self.EDGE and self.horizontal_scan:
91         self.x_direction = -1
92     if self.x_direction < 0 and telemetry["X"] < self.EDGE and
93     ↪ self.horizontal_scan:
94         self.x_direction = 1
95     if self.z_direction > 0 and telemetry["Z"] > self.field_size[0] -
96     ↪ self.EDGE and not self.horizontal_scan:
97         self.z_direction = -1
98     if self.z_direction < 0 and telemetry["Z"] < self.EDGE and not
99     ↪ self.horizontal_scan:
100         self.z_direction = 1
101     return v, r, scan

```

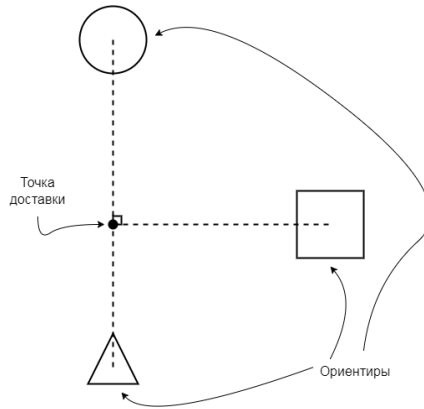
Задача IV.4. Поиск объектов системой технического зрения (30 баллов)

Темы: математическая обработка изображения, геометрия, оптика (анализ изображения), программирование (Python, C++).

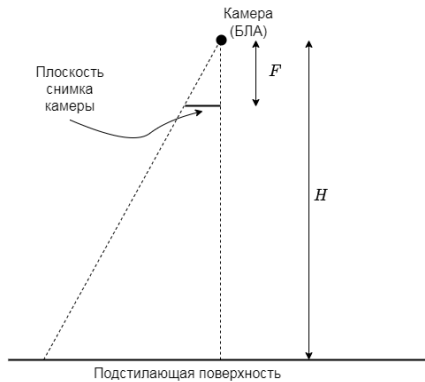
Условие

При разработке систем управления полетом БЛА задача поиска объектов при помощи системы технического зрения нередко является промежуточной. Так, например, при реализации автоматической доставки медицинских грузов точка доставки может задаваться ориентирами, расположенными на местности.

Предположим, что БЛА осуществляет автономный полет и должен рассчитать координаты точки доставки медицинского груза, ориентируясь на измерения системы технического зрения. Известно, что неподалеку от точки доставки расположены три ориентира: круглый, квадратный и треугольный. Точка доставки находится на линии, соединяющей круглый и треугольный ориентиры, на кратчайшем расстоянии от квадратного.



Известны координаты БЛА в момент съемки (x, z, H) , угол его курса, а также параметры камеры: фокусное расстояние F и размер одного пикселя изображения s_p .



По условию задачи камера направлена вертикально вниз, и плоскость камеры параллельна плоскости подстилающей поверхности. Кроме того, вертикальная ось снимка совпадает с продольной осью БЛА.

Напишите программу на языке C++ или Python, реализующую расчет горизонтальных координат местоположения точки доставки на основе анализа фотоснимка, полученного системой технического зрения.

Параметры навигации БЛА на момент съемки:

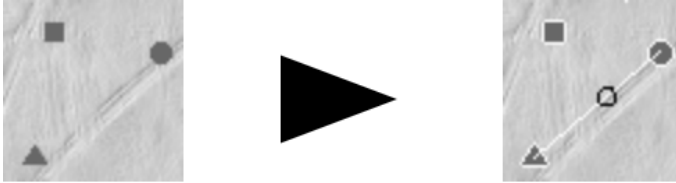
- $x = 400$ (направление на север);
- $z = 250$ (направление на восток);
- $H = 80$;
- $\psi = 30^\circ$ (отсчитывается от севера по часовой стрелке).

Параметры камеры:

- $F = 4,8$ мм;

- $s_p = 15, 2$ мкм.

На платформе Stepik снимок бортовой камеры задается в виде строки в формате PGM изображения размером 65×65 пкс. Гарантируется, что на изображении присутствуют три разных контрастных ориентира. Пример входного изображения и его обработки представлены ниже.



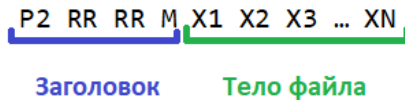
Оценивается точность определения координат точки доставки. При этом усредняется результат, полученный по всем тестам (два открытых и два закрытых). Ошибка определения координат для каждого отдельного теста считается по теореме Пифагора $\Delta = \sqrt{(\Delta x)^2 + (\Delta y)^2}$.

Максимальный балл за задачу — 30. За каждые 0,1 м ошибки относительно истинного местоположения точки доставки снимается 1 балл. Округление результата осуществляется в пользу участников (при величине ошибки 0,099 м, вы получите максимальный балл).

При отправке решения система автоматической проверки покажет вам значения ошибки для каждого из четырех тестов.

Формат входных данных

На вход программы поступает текст PGM-файла, сохраненного в текстовом формате (содержимое PGM файла задается в виде строки через консоль). Текст файла состоит из двух основных частей — заголовка и тела. Описание структуры PGM-файла можно найти по ссылке в списке литературы. Все числа представляют собой целые значения, разделенные пробелами. Тело PGM-файла содержит 4225 значений, описывающих яркость каждого из пикселей изображения размером 65×65 .



Формат выходных данных

Необходимо вывести на стандартный вывод координаты x и z точки сброса груза в м, записанные через пробел.

Список литературы

- Формат PGM [Portable Gray Map]: <http://netpbm.sourceforge.net/doc/pgm.html>.

Тесты

https://disk.yandex.ru/d/VOMgPj-eLv00Fw/Тесты_задача_4.docx

Решение

1. Подключить необходимые библиотеки (`numpy`, `opencv`).
2. Считать содержимое файла PGM.
3. Пользуясь библиотекой `OpenCV`, произвести обработку изображения и выделить контуры ориентиров.
4. Распознать ориентиры на основе количества граней в контурах.
5. Найти координаты центра всех ориентиров на снимке.
6. Найти координаты искомой точки доставки груза на снимке.
7. Найти искомые координаты точки доставки груза путем пересчета координат на снимке в координаты на земле.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```

1 import numpy as np
2 import cv2
3 from math import sqrt, pi, cos, sin
4
5 # Read console input
6 arr = [i for i in input().split()]
7 head = arr[0:4]
8 body = []
9 for i in range(4, len(arr)):
10     body.append(float(arr[i]))
11
12 # Translate input string into numpy array
13 img_bw = np.ones([65,65], dtype=np.uint8)*255
14 for i in range(65):
15     for j in range(65):
16         img_bw[i][j] = body[i * 65 + j]
17
18 # legacy stuff
19 blurred = img_bw
20
21 # setting threshold of gray image
22 _, threshold = cv2.threshold(blurred, 127, 255, cv2.THRESH_BINARY)
23
24 # using a findContours() function
25 _, contours, _ = cv2.findContours(threshold, cv2.RETR_TREE,
26     ↪ cv2.CHAIN_APPROX_SIMPLE)
27
28 # Contours recognition
29 i = 0
30 count = -1
31 x = 0
32 y = 0
33 cx = 0
34 cy = 0

```

```

34 sx = 0
35 sy = 0
36 tx = 0
37 ty = 0
38 # list for storing names of shapes
39 for contour in contours:
40     count += 1
41     # here we are ignoring first counter because
42     # findcontour function detects whole image as shape
43     if i == 0:
44         i = 1
45         continue
46
47     # cv2.approxPloyDP() function to approximate the shape
48     approx = cv2.approxPolyDP(contour, 0.04 * cv2.arcLength(contour, True), True)
49
50     # finding center point of shape
51     M = cv2.moments(contour)
52     if M['m00'] != 0.0:
53         x = int(M['m10']/M['m00'])
54         y = int(M['m01']/M['m00'])
55
56     # putting shape name at center of each shape
57     if len(approx) == 3:
58         #print(f"Triangle center: {x} {y}")
59         tx = x
60         ty = y
61     elif len(approx) == 4:
62         #print(f"Square center: {x} {y}")
63         sx = x
64         sy = y
65     else:
66         #print(f"Circle center: {x} {y}")
67         cx = x
68         cy = y
69
70     # Geometry time!
71     sc = sqrt((cx - sx)**2 + (cy - sy)**2)
72     st = sqrt((tx - sx)**2 + (ty - sy)**2)
73     tc = sqrt((tx - cx)**2 + (ty - cy)**2)
74     p = 0.5 * (sc + st + tc)
75     S = sqrt(p * (p - sc) * (p - st) * (p - tc))
76     h = S * 2 / tc
77     r = sqrt(sc**2 - h**2)
78
79     # Circles intersection
80     A = r**2 - h**2 - cx**2 - cy**2 + sx**2 + sy**2
81     B = A / 2 / (sx - cx)
82     d = (sy - cy) / (sx - cx)
83     E = B**2 - 2 * sx * B + sx**2 - h**2 + sy**2
84     a = d**2 + 1
85     b = 2 * sx * d - 2 * B * d - 2 * sy
86     D = b**2 - 4 * a * E
87     #print(f"Determinant: {D}")
88     y1 = (-b + sqrt(D)) / 2 / a
89     y2 = (-b - sqrt(D)) / 2 / a
90     x1 = B - y1 * d
91     x2 = B - y2 * d
92
93     dirx = tx - cx

```

```
94  diry = ty - cy
95  if np.sign(x1 - cx) == np.sign(dirx) and np.sign(y1 - cy) == np.sign(diry):
96      #print(f"Take first point!")
97      x = x1 + 1
98      y = y1
99  else:
100     x = x2
101     y = y2 + 1
102
103  # Frame transition!
104  heading = 30 * pi / 180.0
105  F = 4.8 * 1e-3
106  sp = 15.2 * 1e-6
107  H = 80
108  dx = H / F * sp
109  shift_x = (x - 65 / 2) * dx
110  shift_y = (65 / 2 - y) * dx
111  geo_x = shift_y * cos(heading) - shift_x * sin(heading) + 400
112  geo_y = shift_y * sin(heading) + shift_x * cos(heading) + 250
113  print(f"{geo_x:.2f} {geo_y:.2f}")
```

Работа наставника НТО при подготовке к заключительному этапу

На этапе подготовки к заключительному этапу НТО наставник решает две важные задачи: помощь участникам в подготовке к предстоящим соревнованиям и формирование устойчивой и слаженной команды. Для подготовки рекомендуется использовать сборники задач прошлых лет. Кроме того, наставнику важно изучить организационные особенности заключительного этапа, чтобы помочь ученикам разобраться в формальных особенностях его проведения.

Наставник НТО также может познакомиться с разработчиками профилей для получения консультации о подготовке к заключительному этапу, дополнительных материалах и способах поддержки высокой мотивации участников.

При работе с командой участников рекомендуется уделить внимание следующим вопросам:

- Сплочение команды. Наставнику необходимо уделить этому особое внимание, если участники команды находятся в разных городах и не имеют возможности встретиться в очном формате. Регулярные встречи, в том числе в дистанционном формате, помогут поддержать эффективную и позитивную коммуникацию внутри команды.
- Анализ состава команды. Необходимо обсудить роли участников в команде и задачи, которые им предстоит решать в рамках выбранных ролей. Кроме того, нужно обсудить взаимозаменяемость ролей.
- Анализ знаний и компетенций участников. Необходимо убедиться, что участники обладают нужными навыками и компетенциями и продумать план по формированию и развитию недостающих навыков и компетенций.
- Составление плана подготовки. График занятий строится, исходя из даты начала заключительного этапа.
- Участие в подготовительных мероприятиях от разработчиков профилей. Перед заключительным этапом проводятся установочные вебинары, разборы задач прошлых лет, практикумы, хакатоны, мастер-классы для финалистов. Информация о таких мероприятиях публикуется в группе НТО в VK и в чатах профилей в Telegram.
- Проведение практикумов или хакатонов. Для этого наставники могут использовать материалы для подготовки к соответствующему профилю и сборники задач прошлых лет. Практикумы и хакатоны могут проводиться дистанционно, рекомендации для этого формата приведены в сборниках 2020–22 гг.

Во время заключительного этапа участников сопровождают модераторы или волонтеры, разработчики профиля и организаторы НТО. Внешнее вмешательство в ход соревнований запрещено. Участники, получившие во время проведения НТО стороннюю помощь, могут быть дисквалифицированы.

Заключительный этап

Предметный тур

Информатика и программирование. 8–11 классы

Тестовые наборы для задач представлены по ссылке — <https://disk.yandex.ru/d/vHU858WP5ugIKw>.

Задача VI.1.1.1. Уборка снега на ВПП (100 баллов)

Имя входного файла: стандартный ввод или `input.txt`.

Имя выходного файла: стандартный вывод или `output.txt`.

Ограничение по времени выполнения программы: 1 с.

Ограничение по памяти: 64 Мбайт.

Условие

За ночь на экспериментальном аэродроме на взлетно-посадочной полосе (ВПП) образовалось много снега. Перед тем как юные участники соревнований будут запускать свои БПЛА самолетного типа, сотрудникам аэропорта была поставлена задача очистить ВПП, благо нового снега не наметает, и вообще погода летная. Очищают ВПП при помощи специальной снегоуборочной техники. В зависимости от ее характеристик за один проход машина убирает с полосы некоторое количество миллиметров снега. Но есть один нюанс: сколько бы ни пыталась машина убрать полностью снег с определенного участка — всегда останется 1 мм, из-за рельефа покрытия ВПП. В любом случае такая высота снежного покрова на ВПП не мешает БПЛА осуществлять взлет и посадку.

Определите наименьшее количество раз, которое понадобится снегоуборочной машине пройти по ВПП для достижения наилучшего возможного результата, и помогите визуально оценить результат уборки снега.

Формат входных данных

На вход программе в первой строке через пробел подаются три целых числа n , m ($1 \leq m \leq 10 \leq n \leq 1000$) — размеры ВПП и k ($1 \leq k \leq 100$) — количество миллиметров снега, которые может снять снегоуборочная машина за один проход по ВПП. Затем в n строках через пробел подаются по m целых чисел $c_{i,j}$ ($0 \leq c_{i,j} \leq 1000$) — максимальная высота снега на участке ВПП.

Формат выходных данных

В первой строке выведите единственное число — минимальное количество проходов, которое необходимо совершить снегоуборочной машине, чтобы достаточным образом очистить ВПП. Далее в n строках по m чисел — высота снега на участках ВПП из входных данных после произведенной уборки.

Примеры

Пример №1

Стандартный ввод
12 5 4
6 6 7 6 6
7 6 6 7 6
7 6 7 6 7
6 7 6 6 7
6 6 6 7 6
7 7 7 7 7
6 6 7 7 7
6 7 7 7 7
6 7 7 6 7
Стандартный вывод
2
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1

Пример программы-решения

Ниже представлено решение на языке Python 3.

```

1 with open('input.txt', 'r') as f:
2     data = list(map(int, f.read().split()))
3     n, m, k = list(map(int, data[:3]))
4     l = list(map(int, data[3:]))
5     maxl = max(l)
6     ans = (max(0, maxl - 1) + k - 1) // k
7     ans_l = ['1' if el else '0' for el in l]
8
9     s = ''
10    s += '{}\n'.format(ans)
11    for i in range(n):
12        s += ' '.join(ans_l[i * m : (i * m) + m]) + '\n'
13
14    with open('output.txt', 'w') as a:
15        a.write(s)

```

Задача VI.1.1.2. Маленькая большая проблема (100 баллов)

Имя входного файла: стандартный ввод или `input.txt`.

Имя выходного файла: стандартный вывод или `output.txt`.

Ограничение по времени выполнения программы: 1 с.

Ограничение по памяти: 64 Мбайт.

Условие

Как известно, больше половины времени программиста занимает отладка программы. Для разработчиков программ управления дронами это утверждение не является исключением. Но есть одна маленькая проблема — из-за того, что на процессоре дрона и компьютере программиста установлены разные архитектуры, порядок записи блоков битов в данных на этих устройствах отличается. На дроне используется *Little-Endian*, в то время как на компьютерах программистов используется *Big-Endian*.

Например, 4-байтное целое число 33686017_{10} будет разложено в следующее двоичное представление:

```
00001000 00000100 00000010 000000012.
```

Такая запись соответствует представлению данных *формата Big-Endian* в памяти устройства. Для того чтобы перевести его в *Little-Endian*, необходимо «развернуть» данные в обратном порядке, то есть итоговая битовая запись будет выглядеть вот так:

```
00000001 00000010 00000100 00001000.
```

Такой формат записи позволяет дрону быстрее производить некоторые операции, например, проверку на четность.

Также довольно часто используются типы целочисленных данных разной длины. Наиболее часто встречаются одно-, двух-, четырех- и восьмибайтные числа. Так как у дронов могут быть различные предназначения и датчики, требуются разные типы целых чисел (включая экзотику в виде трех-, пяти-, шести-, семибайтных) с целью максимально экономичного заполнения памяти.

На предыдущих испытаниях дронов что-то пошло не так, и вам необходимо перевести логи из *формата Little-Endian* в *формат Big-Endian*, чтобы можно было понять причину проблем. Напишите программу, решающую задачу для 1 целого числа.

Формат входных данных

На вход программе в первой строке подается целое число N ($1 \leq N \leq 8$) — количество байтов типа переменной, в которой хранится число. Во второй строке подается целое число X ($-2^{8N-1} \leq X \leq 2^{8N-1} - 1$) — значение целого числа, записанного на дроне.

Формат выходных данных

Целое число — ответ на задачу.

Примеры*Пример №1*

Стандартный ввод
2 15
Стандартный вывод
3840

Пример №2

Стандартный ввод
3 2520118
Стандартный вывод
3568678

Пример №3

Стандартный ввод
5 12639876948
Стандартный вывод
361706680578

Пример №4

Стандартный ввод
2 -8
Стандартный вывод
-1793

Примечание: числа в памяти хранятся в обратном коде. Так, положительные числа в памяти соответствуют двоичному представлению с ведущими нулями. Рассмотрим первый пример: $15_{10} = 1111_2$. Это означает, что двумя байтами число будет представлено следующим образом:

00000000 00001111.

При «развороте» число в памяти будет представлено так:

00001111 00000000,

что соответствует десятичному числу

$$1 \cdot 2^{11} + 1 \cdot 2^{10} + 1 \cdot 2^9 + 1 \cdot 2^8 = 3840.$$

Чтобы арифметические операции производились корректно, отрицательные числа в дополнительном коде должны быть такими, чтобы в сумме с соответствующим по модулю положительным получался ноль (единица, выходящая за границы памяти, игнорируется).

Так, рассмотрим четвертый пример: число $8_{10} = 1000_2$, и в памяти будет храниться как 00000000 00001000. Отрицательное число, соответствующее ему по модулю, будет выглядеть так:

```
11111111 11111000.
```

Если сложить эти два числа, получится 1 00000000 00000000, но ведущая единица выходит за пределы памяти и загирается. После «разворота» число -8 в памяти выглядит так: 11111000 11111111, что соответствует числу

$$0 - 11100000001_2 = (-1 \cdot 2^{10} - 1 \cdot 2^9 - 1 \cdot 2^8 - 1)_{10} = -1793_{10}.$$

Пример программы-решения

Ниже представлено решение на языке C++.

```

1  #include <iostream>
2  using namespace std;
3  int main(){
4      long long x;
5      int n;
6      cin >> n >> x;
7
8      unsigned char bytes[8];
9      unsigned char mask = 0xFF;
10
11     for (int i = 0; i < n; ++i) {
12         bytes[i] = (x >> i * 8) & mask;
13     }
14
15     unsigned long long res = 0;
16
17     for (int i = 0; i < 8; ++i) {
18         res <<= 8;
19         res |= bytes[i];
20     }
21
22     long long ans = res;
23
24     for (int i = 0; i < 8 - n; ++i)
25         ans >>= 8;
26
27     cout << ans << endl;
28 }
```

Задача VI.1.1.3. Рой дронов (100 баллов)

Имя входного файла: стандартный ввод или input.txt.

Имя выходного файла: стандартный вывод или output.txt.

Ограничение по времени выполнения программы: 0,4 с.

Ограничение по памяти: 19,5 Мбайт.

Условие

В воздухе летают дроны в количестве $N < 10^6$ штук ($X_1, X_2, X_3, \dots, X_N$), на земле расположены 3 базовые станции с именами A, B, C , для которых однозначно определено их местоположение. Земля считается абсолютно плоской (2D). В процессе работы дрон устанавливает соединения с базовыми станциями или другими дронами для однозначного определения своего местоположения.

Гарантируется, что:

- никакие два объекта не находятся в одной точке пространства в любой момент времени;
- никакие три объекта не находятся на одной прямой в любой момент времени;
- каждый дрон всегда поддерживает ровно три соединения с тремя разными объектами.

В рамках задачи считается, что местоположение дрона возможно определить однозначно, если местоположение всех объектов, с которыми дрон установил соединение, также возможно определить однозначно. Необходимо выяснить, определяется ли однозначно местоположение каждого дрона и количество таких дронов.

Формат входных данных

В первой строке передается одно число N ($1 \leq N \leq 100009$) — количество работающих дронов.

Далее в следующих $3 \cdot N$ строках находятся пары объектов F, T , означающие, что дрон F установил соединение с объектом T .

Формат выходных данных

Выведите в первой строке количество дронов N , местоположение которых возможно определить однозначно, а во второй строке — имена самих дронов в любом порядке через пробел. Если таковых дронов нет, выведите только в первой строке «0».

Примеры

Пример №1

Стандартный ввод
1
X1 A
X1 B
X1 C
Стандартный вывод
1
X1

Пример №2

Стандартный ввод
2 X1 A X1 B X1 C X2 X1 X2 B X2 C
Стандартный вывод
2 X1 X2

Пример программы-решения

Ниже представлено решение на языке C++.

```

1  #include <iostream>
2  #include <string>
3  #include <vector>
4  #include <map>
5  using namespace std;
6
7  map<string,vector<string>> con;
8  map<string,int> pl = {"A",1},{"B",1},{"C",1};
9  int discon=0;
10
11 void dfs(string dron) {
12     pl[dron] = 0;
13     int k = 0;
14     for (auto nxt:con[dron]) {
15         if (pl[nxt]==-1)
16             dfs(nxt);
17         k+=pl[nxt];
18     }
19     if (k==3)
20         pl[dron]=1;
21     else
22         discon++;
23 }
24
25 int main() {
26     int n;
27     cin>>n;
28     for (int i=0;i<3*n;i++) {
29         string s1,s2;
30         cin>>s1>>s2;
31         con[s1].emplace_back(s2);
32         pl[s1]=-1;
33     }
34     for (auto p : pl)
35         if (p.second==-1)
36             dfs(p.first);
37
38     pl.erase("A");

```

```
39     pl.erase("B");
40     pl.erase("C");
41
42     cout << n - discon << endl;
43
44     if (n - discon > 0) {
45         for (auto p : pl)
46             if (p.second == 1) cout << p.first << ' ';
47         cout << endl;
48     }
49
50     return 0;
51 }
```

Задача VI.1.1.4. Покрышка колес шасси (100 баллов)

Имя входного файла: стандартный ввод или `input.txt`.

Имя выходного файла: стандартный вывод или `output.txt`.

Ограничение по времени выполнения программы: 1 с.

Ограничение по памяти: 128 Мбайт.

Условие

Проводить техобслуживание колес шасси — достаточно скучное дело. В перерыве работник решил нанести белым маркером на покрышке некоторые числа и запустил вращение колеса. Колесо вращалось настолько медленно, что можно было переписывать числа в тетрадь, чем и занялся юноша. Так, в момент остановки колеса выяснилось, что оно находится в том же состоянии, где было начато движение. По записи из тетради определите, какое наименьшее количество чисел могло быть на покрышке.

Формат входных данных

В первой строке подается натуральное число N , не превосходящее 10^6 . В следующей строке подается N чисел, записанных через пробел. Первое число всегда совпадает с последним. Все числа в тетради — натуральные, не превышающие 32000.

Формат выходных данных

Выведите одно число — ответ на задачу.

*Примеры**Пример №1*

Стандартный ввод
15
1 2 1 3 1 2 1 4 1 2 1 3 1 2 1
Стандартный вывод
14

Пример №2

Стандартный ввод
13
1 2 1 1 2 1 1 2 1 1 2 1 1
Стандартный вывод
3

Пример №3

Стандартный ввод
12
1 2 1 1 2 1 1 2 1 1 2 1
Стандартный вывод
11

Пример программы-решения

Ниже представлено решение на языке Python 3.

```

1 x = input()
2 x = [int(t) for t in input().split()]
3 x.pop()
4 n = len(x)
5
6 for i in range(1, n + 1):
7     if n % i == 0:
8         ok = True
9
10        for el in range(i):
11            if not(ok):
12                break
13
14            for rep in range(1, n // i):
15                if x[el] != x[rep * i + el]:
16                    ok = False
17                    break
18
19            if ok:
20                break
21
22 print(i)

```

Задача VI.1.1.5. Покрышка колес шасси — 2 (100 баллов)

Имя входного файла: стандартный ввод или input.txt.

Имя выходного файла: стандартный вывод или output.txt.

Ограничение по времени выполнения программы: 5 с.

Ограничение по памяти: 64 Мбайт.

Условие

В предыдущей задаче по записи из тетради вы определили, какое наименьшее количество чисел могло быть на покрышке, и на основе этого восстановили, в каком порядке записаны числа. Теперь предлагаем среди всех циклических сдвигов $P_N = P$ найти наименьший (ту последовательность, которая является наименьшей среди остальных).

Наименьшей считается такая последовательность P_a , у которой для всех других последовательностей $P_{b \neq a}$ или $P_b = P_a$, или существует такой i_b , что $P_a[i_b] < P_b[i_b]$ и для всех $j < i_b$ справедливо $P_a[j] = P_b[j]$.

Формат входных данных

В первой строке подается целое число N ($1 \leq N \leq 10^5$) — количество чисел в последовательности. В следующей строке подаются N чисел, записанных через пробел. Первое число всегда совпадает с последним. Все числа в тетради — натуральные, не превышающие 32000.

Формат выходных данных

Через пробел выведите последовательность чисел — ответ на задачу.

Примеры

Пример №1

Стандартный ввод
5
4 3 4 2 4
Стандартный вывод
2 4 4 3 4

Пример №2

Стандартный ввод
3
1 2 1
Стандартный вывод
1 1 2

Пример №3

Стандартный ввод
4
2 1 2 1
Стандартный вывод
1 2 1 2

Примечание: принцип выбора наименьшей последовательности чисел аналогичен выбору лексикографически наименьшей последовательности букв.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```

1 from itertools import zip_longest, islice
2 def suffix_matrix_stanford(s):
3     n = len(s)
4     k = 1
5     line = s
6     ans = list(line)
7     while k < n:
8         L = sorted((key, index) for index, key in enumerate(zip_longest(line,
9             ↪ islice(line, k, None), fillvalue=-1)))
10        line = [0] * n
11        for i in range(n):
12            line[L[i][1]] = (line[L[i - 1][1]] if i and L[i][0] == L[i - 1][0]
13                ↪ else i)
14        ans = line
15        k <<= 1
16    return ans
17 def solve2(dataset):
18     data = [int(e) for e in dataset.split()]
19     n = data[0]
20     s = data[1:]
21
22     ns = s[:n] + s[:n-1]
23     p = suffix_matrix_stanford(ns)
24     minp = min(p[:n])
25     pos = 0
26
27     for i in range(n):
28         if p[i] == minp:
29             pos = i
30             break
31
32     return ' '.join(str(i) for i in ns[pos : pos + n])
33
34 test = ' '
35 with open('input.txt', 'r') as f:
36     test = f.read()
37
38 sol = solve2(test)
39 with open('output.txt', 'w') as fo:
40     fo.write(sol)

```

Физика. 8–9 классы

Таблица VI.1.1: Таблица постоянных величин

Постоянные величины
Гравитационная постоянная $G = 6,67 \times 10^{-11} \text{ Н}\cdot\text{м}^2/\text{кг}^2$
Универсальная газовая постоянная $R = 8,31 \text{ Дж}/(\text{моль}\cdot\text{К})$
Электрическая постоянная $\varepsilon_0 = 8,85 \times 10^{-12} \text{ Ф/м}$
Средний радиус Земли $R = 6,4 \times 10^6 \text{ м}$
Масса Земли $M = 5,972 \times 10^{24} \text{ кг}$
Ускорение свободного падения $g = 10 \text{ м/с}^2$

Задача VI.1.2.1. Когда встречаются стрелки (20 баллов)

Условие



Одним из самых важных приборов, используемых в авиационных полетах, являются часы, на которых есть не только часовая, а также минутная и секундная стрелки. Определите промежуток времени, через который встречаются минутная и часовая стрелки.

Критерии оценивания

Задачи должны быть оформлены согласно ФГОС, а их решение — демонстрировать знание законов и способы их применения в конкретной ситуации. В решении задачи должны быть записаны положения теории и физические законы, закономерности, применение которых необходимо для решения задачи; представлены и описаны вновь введенные величины; представлены необходимые математические преобразования и расчеты (подстановка числовых данных в конечную формулу); представлен правильный ответ с указанием единиц измерения.

1. Правильно определено соотношение для расчета скорости конца часовой стрелки — 5 баллов.
2. Правильно определено соотношение для расчета скорости конца минутной стрелки — 5 баллов.
3. Получено выражение для вычисления времени встречи — 5 баллов.
4. Правильно проведены вычисления — 5 баллов.

5. Если решения нет, но приведены разумные рассуждения в направлении решения — 5 баллов.

Решение

$$\omega_{\text{мин}} = \frac{360^\circ}{T_{\text{мин}}}, \quad \omega_{\text{час}} = \frac{360^\circ}{T_{\text{час}}}.$$

За исследуемый промежуток времени t стрелки опишут угол:

$$\varphi_{\text{час}} = \omega_{\text{час}} t = \frac{360^\circ}{T_{\text{час}}} t,$$

$$360^\circ + \varphi_{\text{час}} = \omega_{\text{мин}} t = \frac{360^\circ}{T_{\text{мин}}} t.$$

Вычитаем из первого второе:

$$-360^\circ = \frac{360^\circ}{T_{\text{час}}} t - \frac{360^\circ}{T_{\text{мин}}} t,$$

$$\frac{1}{T_{\text{мин}}} - \frac{1}{T_{\text{час}}} = \frac{1}{t}.$$

Ответ: 1,09 ч.

Задача VI.1.2.2. Космический лифт (30 баллов)

Условие

В последнее время все чаще можно слышать информацию о гипотезе создания космического лифта, с помощью которого осуществляется заброска грузов в открытый космос без использования ракет. Есть также предложения о создании троса, в основе которого лежат углеродные нанотрубки графена, масса 1 см^3 этого вещества составляет 0,16 мг. Оцените длину струны площадью 1 мм^2 и ее массу.

Критерии оценивания

Задачи должны быть оформлены согласно ФГОС, а их решение демонстрировать знание законов и способы их применения в конкретной ситуации. В решении задачи должны быть записаны положения теории и физические законы, закономерности, применение которых необходимо для решения задачи; представлены и описаны вновь введенные величины; представлены необходимые математические преобразования и расчеты (подстановка числовых данных в конечную формулу); представлен правильный ответ с указанием единиц измерения.

1. Правильно определено соотношение для расчета первой космической скорости — 5 баллов.
2. Правильно определено соотношение для расчета скорости движения по стационарной орбите — 5 баллов.
3. Правильно определено соотношение для расчета длины троса — 10 баллов.

4. Правильно определено соотношение для расчета массы — 5 баллов.
5. Правильно проведены вычисления — 5 баллов.
6. Если решения нет, но приведены разумные рассуждения в направлении решения — 5 баллов.

Решение

$$v = \sqrt{G \frac{M}{R+l}}, \quad v = \frac{2\pi(R+l)}{T}.$$

Приравниваем и находим

$$R+l = 42,07 \cdot 10^6 \text{ м.}$$

$$l = 35,67 \cdot 10^6 \text{ м.}$$

$$m = \rho V = \rho S l = 0,16 \cdot 10^{-6} \cdot 35,67 \cdot 10^6 = 5,7 \text{ кг.}$$

Ответ: $35,67 \cdot 10^6$ м; 5,7 кг.

Задача VI.1.2.3. Горячая еда (25 баллов)

Условие

При полете на самолетах пассажирам предлагается горячая пища в специальных контейнерах (алюминиевых касалетках), которая разогревается до 230°C . В печи есть три режима. В первом работает один ТЭН, во втором — два ТЭНа, включенные последовательно, в третьем режиме — параллельно. Напряжение на борту 200 В, сопротивление каждого ТЭНа 20 Ом. Определите, при каком подключении приборов время разогревания пищи составит от 5 до 7 мин. В печь входит 12 порций по 250 г. Масса касалетки 10 г. Удельную теплоемкость пищи принять равной $3000 \text{ Дж}/(\text{кг}\cdot^\circ\text{C})$, алюминия $920 \text{ Дж}/(\text{кг}\cdot^\circ\text{C})$, коэффициент полезного действия печи 80%, температура в салоне 20°C .

Критерии оценивания

Задачи должны быть оформлены согласно ФГОС, а их решение демонстрировать знание законов и способы их применения в конкретной ситуации. В решении задачи должны быть записаны положения теории и физические законы, закономерности, применение которых необходимо для решения задачи; представлены и описаны вновь введенные величины; представлены необходимые математические преобразования и расчеты (подстановка числовых данных в конечную формулу); представлен правильный ответ с указанием единиц измерения.

1. Правильно определено соотношение для расчета количества теплоты — 5 баллов.
2. Правильно определено соотношение для расчета массы — 5 баллов.
3. Правильно определено соотношение для расчета работы тока — 5 баллов.
4. Правильно определено соотношение для расчета времени — 5 баллов.

5. Правильно проведены вычисления — 5 баллов.
6. Если решения нет, но приведены разумные рассуждения в направлении решения — 5 баллов.

Решение

$$Q = c_{\text{пищ}} m_{\text{пищ}} \Delta t + c_{\text{ал}} m_{\text{ал}} \Delta t.$$

$$Q_1 = \frac{U^2}{R} \tau_1, \quad Q_2 = \frac{U^2}{2R} \tau_2, \quad Q_3 = 2 \frac{U^2}{R} \tau_3.$$

$$\tau_1 = \frac{\text{КПД}_{12} (c_{\text{пищ}} m_{\text{пищ}} \Delta t + c_{\text{ал}} m_{\text{ал}} \Delta t)}{\frac{U^2}{R}} = 12,75 \text{ мин.}$$

$$\tau_2 = \frac{\text{КПД}_{12} (c_{\text{пищ}} m_{\text{пищ}} \Delta t + c_{\text{ал}} m_{\text{ал}} \Delta t)}{\frac{U^2}{2R}} = 35,5 \text{ мин.}$$

$$\tau_3 = \frac{\text{КПД}_{12} (c_{\text{пищ}} m_{\text{пищ}} \Delta t + c_{\text{ал}} m_{\text{ал}} \Delta t)}{2 \frac{U^2}{R}} = 6,4 \text{ мин.}$$

Ответ: время нагрева при параллельном включении составляет 6,4 мин.

Задача VI.1.2.4. Авиационное топливо (25 баллов)

Условие

Одно из направлений развития авиастроения связано с увеличением мощности летательных аппаратов, которая напрямую зависит от вида топлива, используемого в двигателях. Повышение эффективности горючего может привести к возгоранию в рабочих режимах, поэтому в авиационный бензин добавляются детонационные жидкости, например, бензол. Пусть цилиндрическая емкость высотой 50 см до верха заполнена равными массами бензина (плотность $0,76 \text{ г/см}^3$) и бензола (плотность 876 кг/м^3). Определите давление жидкостей на дно сосуда при условии, что жидкости не смешиваются. Ответ округлите до целых.

Критерии оценивания

Задачи должны быть оформлены согласно ФГОС, а их решение демонстрировать знание законов и способы их применения в конкретной ситуации. В решении задачи должны быть записаны положения теории и физические законы, закономерности, применение которых необходимо для решения задачи; представлены и описаны вновь введенные величины; представлены необходимые математические преобразования и расчеты (подстановка числовых данных в конечную формулу); представлен правильный ответ с указанием единиц измерения.

1. Правильно записано базовое выражение для определения массы — 5 баллов.
2. Правильно записано уравнение гидростатики — 10 баллов.
3. Правильно проведены вычисления — 10 баллов.
4. Если решения нет, но приведены разумные рассуждения в направлении решения — 5 баллов.

Решение

$m_1 = m_2$, следовательно: $\rho_1 g h_1 S = \rho_2 g h_2 S$, тогда:

$$\rho_1 h_1 = \rho_2 h_2,$$

$$h = h_1 + h_2.$$

Значит:

$$h_1 = h \frac{\rho_2}{\rho_1 + \rho_2},$$

$$h_2 = h \frac{\rho_1}{\rho_1 + \rho_2}.$$

С другой стороны:

$$p = \rho_1 g h_1 + \rho_2 g h_2,$$

$$p = g h \frac{2\rho_1 \rho_2}{\rho_1 + \rho_2},$$

$$p = 4069 \text{ Па}.$$

Ответ: 4069 Па.

Физика. 10–11 классы

Таблица VI.1.2: Таблица постоянных величин

Постоянные величины
Гравитационная постоянная $G = 6,67 \times 10^{-11} \text{ Н}\cdot\text{м}^2/\text{кг}^2$
Универсальная газовая постоянная $R = 8,31 \text{ Дж}/(\text{моль}\cdot\text{К})$
Электрическая постоянная $\epsilon_0 = 8,85 \times 10^{-12} \text{ Ф}/\text{м}$
Средний радиус Земли $R = 6,4 \times 10^6 \text{ м}$
Масса Земли $M = 5,972 \times 10^{24} \text{ кг}$
Ускорение свободного падения $g = 10 \text{ м}/\text{с}^2$

Задача VI.1.3.1. Мониторинг окружающей среды (15 баллов)**Условие**

Метеозонды являются основным источником сведений о погодных условиях, перемещениях воздушных масс, формировании циклонов и антициклонов. Благодаря полученным данным, в режиме реального времени создается компьютерный прогноз погоды на Земном шаре. Высота подъема этих устройств достигает 35 км. Для мониторинга состояния окружающей среды с метеозонда, который двигается равномерно вверх, на некоторой высоте в горизонтальном направлении была отстрелена капсула с оборудованием со скоростью 10 м/с. Определите, через какой промежуток времени капсула окажется на Земле, если в момент ее падения зонд оказался на высоте 245 м.

Критерии оценивания

Задачи должны быть оформлены согласно ФГОС, а их решение демонстрировать знание законов и способы их применения в конкретной ситуации. В решении задачи должны быть записаны положения теории и физические законы, закономерности, применение которых необходимо для решения задачи; представлены и описаны вновь введенные величины; представлены необходимые математические преобразования и расчеты (подстановка числовых данных в конечную формулу); представлен правильный ответ с указанием единиц измерения.

1. Правильно записано уравнение движения груза — 5 баллов.
2. Правильно записано уравнение движения шара — 5 баллов.
3. Правильно проведены вычисления — 5 баллов.
4. Если решения нет, но приведены разумные рассуждения в направлении решения — 5 баллов.

Решение

Уравнение движения груза: $y_k = h + v_0 t - gt^2/2$.

Уравнение движения шара: $y_{ш} = h + v_0 t$, где

v_0 — скорость шара и начальная скорость груза;

h — высота, с которой брошен груз.

Через искомый промежуток времени $y_k = 0$ м и $y_{ш} = 245$ м:

$$0 = h + v_0 t - gt^2/2,$$

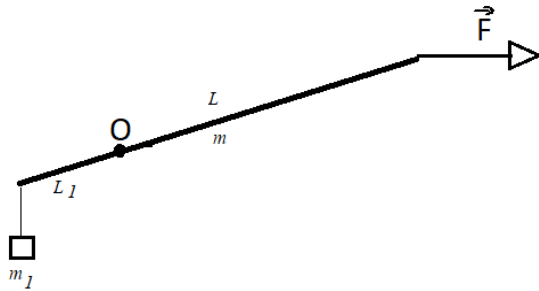
$$245 = h + v_0 t.$$

Ответ: 7 с.

Задача VI.1.3.2. Крылья квадрокоптера (20 баллов)

Условие

При испытании модели крыла квадрокоптера длиной 0,5 м и массой 300 г его устанавливают на стенде так, что оно вращается вокруг оси, расположенной на расстоянии 0,1 м от его короткого конца, на который подвешивают груз массой 600 г. Какой будет угол наклона крыла к горизонту, если на другой его конец действует горизонтальная сила 5 Н? Считать крыло однородным стержнем. Ответ предоставить в градусах и округлить до целых.

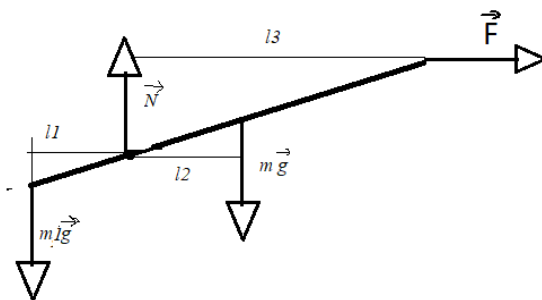


Критерии оценивания

Задачи должны быть оформлены согласно ФГОС, а их решение демонстрировать знание законов и способы их применения в конкретной ситуации. В решении задачи должны быть записаны положения теории и физические законы, закономерности, применение которых необходимо для решения задачи; представлены и описаны вновь введенные величины; представлены необходимые математические преобразования и расчеты (подстановка числовых данных в конечную формулу); представлен правильный ответ с указанием единиц измерения.

1. Правильно записано условие равновесия — 10 баллов.
2. Правильно определены плечи сил — 5 баллов.
3. Правильно проведены вычисления — 5 баллов.
4. Если решения нет, но приведены разумные рассуждения в направлении решения — 5 баллов.

Решение



Условие равновесия крыла длиной $L = 0,5$ м, для правила моментов:

$$mgl_1 + Fl_3 - m_1gl + N \cdot 0 = 0, \text{ где}$$

$$l_1 = \left(\frac{L}{2} - L_1 \right) \cos \alpha,$$

$$\begin{aligned}
 l_2 &= L_1 \cos \alpha, \\
 l_3 &= (L - L_1) \sin \alpha, \\
 F(L - L_1) \sin \alpha + mg \left(\frac{L}{2} - L_1 \right) \cos \alpha - m_1 g L_1 \cos \alpha &= 0, \\
 \operatorname{tg} \alpha &= \frac{g \left(m_1 L_1 - m \left(\frac{L}{2} - L_1 \right) \right)}{(L - L_1) F}, \\
 \operatorname{tg} \alpha &= 0,075, \\
 \alpha &\approx 4^\circ.
 \end{aligned}$$

Ответ: 4° .

Задача VI.1.3.3. Авиационное топливо (20 баллов)

Условие

Одно из направлений развития авиастроения связано с увеличением мощности летательных аппаратов, которая напрямую зависит от вида топлива, используемого в двигателях. Повышение эффективности горючего может привести к возгоранию в рабочих режимах, поэтому в авиационный бензин добавляются детонационные жидкости, например, бензол. Пусть цилиндрическая емкость высотой 50 см до верха заполнена равными массами бензина (плотность $0,76 \text{ г/см}^3$) и бензола (плотность 876 кг/м^3). Определите суммарное давление жидкостей на дно сосуда при условии, что жидкости не смешиваются. Ответ округлите до целых.

Критерии оценивания

Задачи должны быть оформлены согласно ФГОС, а их решение демонстрировать знание законов и способы их применения в конкретной ситуации. В решении задачи должны быть записаны положения теории и физические законы, закономерности, применение которых необходимо для решения задачи; представлены и описаны вновь введенные величины; представлены необходимые математические преобразования и расчеты (подстановка числовых данных в конечную формулу); представлен правильный ответ с указанием единиц измерения.

1. Правильно записано базовое выражение для определения массы — 5 баллов.
2. Правильно записано уравнение гидростатики — 5 баллов.
3. Правильно проведены вычисления — 10 баллов.
4. Если решения нет, но приведены разумные рассуждения в направлении решения — 5 баллов.

Решение

$m_1 = m_2$, следовательно: $\rho_1 g h_1 S = \rho_2 g h_2 S$, тогда:

$$\rho_1 h_1 = \rho_2 h_2,$$

$$h = h_1 + h_2.$$

Значит:

$$h_1 = h \frac{\rho_2}{\rho_1 + \rho_2},$$

$$h_2 = h \frac{\rho_1}{\rho_1 + \rho_2}.$$

С другой стороны:

$$p = \rho_1 g h_1 + \rho_2 g h_2,$$

$$p = g h \frac{2\rho_1\rho_2}{\rho_1 + \rho_2},$$

$$p = 4069 \text{ Па}.$$

Ответ: 4069 Па.

Задача VI.1.3.4. Система электробезопасности (20 баллов)

Условие

В системе электробезопасности летательного аппарата используется участок цепи, представленный на рисунке.



Общая электрическая емкость плоских бумаго-масляных конденсаторов равна 0,01 мкФ, входное напряжение 60 В. Определите значение силы тока утечки, если диэлектрическая проницаемость заполнения конденсатора равна 2,2, а его удельное сопротивление составляет 10^{12} Ом·м.

Критерии оценивания

Задачи должны быть оформлены согласно ФГОС, а их решение демонстрировать знание законов и способы их применения в конкретной ситуации. В решении задачи должны быть записаны положения теории и физические законы, закономерности, применение которых необходимо для решения задачи; представлены и описаны вновь введенные величины; представлены необходимые математические преобразования и расчеты (подстановка числовых данных в конечную формулу); представлен правильный ответ с указанием единиц измерения.

1. Правильно записан закон Ома для участка цепи — 5 баллов.
2. Правильно записано выражение для расчета емкости последовательного соединения конденсаторов — 5 баллов.
3. Правильно записано выражение для расчета сопротивления последовательного соединения резисторов — 5 баллов.
4. Правильно проведены вычисления — 5 баллов.

5. Если решения нет, но приведены разумные рассуждения в направлении решения — 5 баллов.

Решение

1. По закону Ома для участка цепи:

$$I = \frac{U}{R}.$$

2. Учитывая, что электрическая емкость плоского конденсатора:

$$C = \frac{\varepsilon\varepsilon_0 S}{d},$$

а также последовательное соединение конденсаторов:

$$C = \frac{C_1 C_2}{C_1 + C_2} = \frac{\varepsilon\varepsilon_0 S_1 S_2}{S_1 d_2 + S_2 d_1}. \quad (\text{VI.1.1})$$

3. Сопротивление связано с параметрами диэлектрического заполнения:

$$R = \frac{\rho l}{S}.$$

Учитывая последовательное соединение:

$$R = R_1 + R_2 = \rho \frac{S_1 d_2 + S_2 d_1}{S_1 S_2}. \quad (\text{VI.1.2})$$

Сопоставляем (VI.1.1) и (VI.1.2) и получаем:

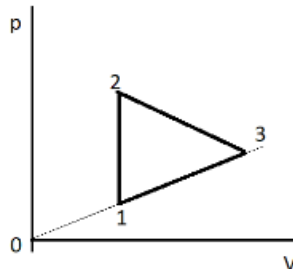
$$R = \rho \frac{\varepsilon\varepsilon_0}{C}.$$

Ответ: 0,03 мкА.

Задача VI.1.3.5. Тяжелая работа (25 баллов)

Условие

В испытательном образце двигателя один моль идеального газа последовательно проходит три этапа замкнутого цикла 1–2–3. Начальная температура газа равна T_1 , в точках 2 и 3 температуры равны T_{23} . Определите работу цикла.



Критерии оценивания

Задачи должны быть оформлены согласно ФГОС, а их решение демонстрировать знание законов и способы их применения в конкретной ситуации. В решении задачи должны быть записаны положения теории и физические законы, закономерности, применение которых необходимо для решения задачи; представлены и описаны вновь введенные величины; представлены необходимые математические преобразования и расчеты (подстановка числовых данных в конечную формулу); представлен правильный ответ с указанием единиц измерения.

1. Правильно записаны газовые законы — 5 баллов.
2. Правильно записано выражение для расчета работы — 5 баллов.
3. Правильно записано соотношение давления и объема на участке 1–2 графика — 5 баллов.
4. Правильно получена формула для расчета работы при увеличении объема — 5 баллов.
5. Правильно получена формула для расчета работы при уменьшении объема — 5 баллов.
6. Если решения нет, но приведены разумные рассуждения в направлении решения — 5 баллов.

Решение

Участок 1–2 соответствует изохорному процессу, поэтому $A_{12} = 0$.

Используя график, находим работу на 2–3, как площадь трапеции:

$$A_{23} = \frac{p_2 + p_3}{2}(V_3 - V_2).$$

Из уравнения Менделеева для одного моля газа: $p_2V_2 = p_3V_3 = RT_{23}$.

Следовательно:

$$A = A_{23} - A_{31},$$

$$A_{23} = RT_{23} \frac{(V_3/V_2)^2 - 1}{2(V_3/V_2)}.$$

Составим систему уравнений.

Исходя из рисунка:

$$\frac{p_1}{V_1} = \frac{p_3}{V_3}.$$

Исходя из закона Шарля для участка 1–2:

$$\frac{p_1}{T_1} = \frac{p_2}{T_2}.$$

Исходя из закона Бойля – Мариотта для участка 2–3:

$$p_2V_2 = p_3V_3.$$

А также, учитывая $V_1 = V_2$ и $T_2 = T_3 = T_{23}$. Получим:

$$\frac{V_3^2}{V_2^2} = \frac{T_{23}}{T_1},$$

$$A_{23} = RT_{23} \frac{T_{23}/T_1 - 1}{2\sqrt{T_{23}/T_1}} = \frac{R}{2} \sqrt{\frac{T_{23}}{T_1}} (T_{23} - T_1),$$
$$A_{31} = \frac{p_1 + p_3}{2} (V_3 - V_1) = \frac{p_3 V_3 - p_1 V_1}{2} = \frac{R}{2} (T_{23} - T_1),$$
$$A = \frac{R}{2} \left(\sqrt{\frac{T_{23}}{T_1}} - 1 \right) (T_{23} - T_1).$$

Ответ: $A = \frac{R}{2} \left(\sqrt{\frac{T_{23}}{T_1}} - 1 \right) (T_{23} - T_1).$

Инженерный тур

Общая информация

Аэрологистика медицинского оборудования и медикаментов для экстренной медицинской помощи с помощью БЛА самолетного типа.

Легенда задачи

Рассматривается ситуация нехватки медицинского оборудования и/или медикаментов в труднодоступных местах — например, группа туристов в горах или в лесу попала в беду или жилец одиночного дома, расположенного в труднодоступной местности на большом удалении от крупного населенного пункта, испытывает потребность в медикаментах. Цель — оперативно в автоматическом режиме с применением БЛА самолетного типа доставить медицинское оборудование и медикаменты до адресата.

Задача БЛА — в автоматическом режиме в условиях возможного возникновения отказов осуществить доставку медицинского оборудования и медикаментов нескольким адресатам за ограниченное время, используя для окончательного определения координат места доставки техническое зрение.

Участники на финале работают не только с симулятором полета, но и с реальными датчиками, органами управления и двигательной установкой БЛА самолетного типа.

Требования к команде и компетенциям участников

Количество участников в команде: 3–4 человека.

Компетенции членов команды (роли, которые должны быть представлены в команде):

1. капитан;
2. программист C++ и Python;
3. физик, математик;
4. электронщик.

Оборудование и программное обеспечение

Наименование	Описание
Симулятор полета UAViant	Используется для отработки решений участников в виртуальной среде

Наименование	Описание
Стенд полунатурного моделирования	Используется для отработки алгоритмов управления БЛА
Комплект для пайки	Используется для пайки необходимых компонент на плату управления БЛА
Плата управления БЛА для симулятора UAViant на базе контроллера ESP32 с USB-кабелем	Используется для отработки решений участников
Программатор ESP-Prog	Используется для внутренней пошаговой отладки программы с применением брэйк-поинтов
БЛА Skywalker Titan	Используется для отработки алгоритмов управления БЛА
Языки программирования C++ и Python	Используются для написания программного кода во всех задачах финала

Задачи

Задача VI.2.5.1.

Для реализации алгоритмов автоматического управления полетом БЛА необходимо как можно точнее определять его навигационные параметры в каждый момент времени.

В современных бортовых измерительных системах используются алгоритмы комплексной обработки информации, позволяющие совместно обрабатывать измерения разных датчиков для повышения точности навигационного решения.

Условие

1. Припаяйте контактные колодки на печатную плату и соберите блок, присоединив к печатной плате плату с контроллером ESP32 и плату с датчиком MPU6050.
2. Напишите программу для ESP32, реализующую расчет угла тангажа по измерениям акселерометра и гироскопа с использованием комплементарной фильтрации. Для повышения точности определения угла тангажа необходимо провести начальную калибровку датчиков.

Формат входных данных

Печатная плата, контактные колодки, принадлежности для пайки, датчик MPU6050 и плата ESP32.

Формат выходных данных

Печатная плата с напаянными контактными колодками, программа на Arduino, реализующая расчет угла тангажа в градусах по измерениям гироскопа и акселерометра.

Критерии оценивания

- Оценка пайки: 0,5 балла — пайка необходимых компонент; 0,5 балла — аккуратность пайки.
- Для оценки точности расчета угла датчик последовательно поворачивается на углы тангажа -5° , 10° , 20° и 10° , 0° , -15° .
- Оценивается средняя ошибка определения угла тангажа.
- Если величина ошибки составляет меньше 2° , за решение задачи начисляется 4 балла.
- За каждый дополнительный 1° (полный) ошибки снимается 1 балл.
- Правильный ответ на вопрос по теории прибавляет 1 балл (всего два вопроса).

Решение

Вариант алгоритма решения задачи:

1. Припаять контактные колодки к печатной плате.
 2. Подключить датчик MPU6050 к припаянным колодкам на печатной плате.
 3. Подключить плату ESP32 WROOM к припаянным колодкам печатной плате.
 4. Определить сдвиг нуля и масштабные коэффициенты акселерометра.
 5. Определить сдвиг нуля гироскопа.
 6. Написать код программы расчета углов крена и тангажа.
 7. Подобрать весовой коэффициент фильтра для получения наибольшей точности.
 8. Проверить точность определения углов в ходе эксперимента.
- Пункты 7 и 8 повторять до достижения необходимой точности.

Пример программы-решения

Ниже представлено решение на языке C++.

```

1  #include "Arduino.h"
2  #include <Adafruit_MPU6050.h>
3  #include <Adafruit_Sensor.h>
4  #include <Wire.h>
5  Adafruit_MPU6050 mpu;
6
7  float angleRangeKeeper(float angle)
8  {
9      angle = fmod(angle, 360.0);
10     if (angle > 180) {
11         angle -= 360.0;
12     }
13     else if (angle <= -180) {
14         angle += 360.0;
15     }
16     return angle;
17 }
18
19 void setup(void) {
20     Serial.begin(115200);

```

```

21     while (!Serial)
22         delay(10);
23
24     Serial.println("Adafruit MPU6050 test!");
25     //Try to initialize!
26     if (!mpu.begin()) {
27         Serial.println("Failed to find MPU6050 chip");
28         while (1) {
29             delay(10);
30         }
31     }
32     Serial.println("MPU6050 Found!");
33     mpu.setAccelerometerRange(MPU6050_RANGE_8_G);
34     mpu.setGyroRange(MPU6050_RANGE_500_DEG);
35     mpu.setFilterBandwidth(MPU6050_BAND_21_HZ);
36     Serial.println("");
37     delay(100);
38 }
39
40 float pitchg = 0.0;
41 float pitcha = 0.0;
42
43 float acc[] = {0.2, 0.2, 0.08};
44 float gyr[] = {0.02, 0, 0.01};
45 float acc_scale = 1.03;
46
47 void loop() {
48     /* Get new sensor events with the readings */
49     sensors_event_t a, g, temp;
50     mpu.getEvent(&a, &g, &temp);
51     float gyr_x = g.gyro.x + gyr[0];
52     float gyr_z = g.gyro.z + gyr[2];
53
54     float acc_z = (a.acceleration.z + acc[2]) * acc_scale;
55     float acc_y = (a.acceleration.y + acc[1]) * acc_scale;
56     pitcha = atan2(acc_y, acc_z) * 180.0 / 3.1415;
57     if (abs(gyr_x) > 0.07)
58     {
59         pitchg += angleRangeKeeper(gyr_x * 0.2 * 180.0 / 3.1415);
60     }
61
62     float k = 0.3;
63     float Pitch_deg = angleRangeKeeper(k * pitchg + (1 - k) * pitcha);
64     Serial.print("Angle Pitch: ");
65     Serial.print(pitcha);
66     Serial.print(" + ");
67     Serial.print(pitchg);
68     Serial.print(" = ");
69     Serial.println(-Pitch_deg);
70     delay(200);
71 }

```

Материалы для подготовки

- MPU6050 Arduino: <https://alexgyver.ru/arduino-mpu6050/>.

Задача VI.2.5.2.

Одной из базовых функций, реализуемых БЛА, является автономный полет по заданной траектории. Для осуществления такого полета разрабатываются специальные системы стабилизации, способные регулировать высоту и направление полета БЛА.

Условие

Разработайте программу, реализующую автоматическое управление углом крена и тангажа БЛА для осуществления его движения с заданным курсом и эшелонем.

Формат входных данных

Заданный эшелон и значение угла курса, система автоматической стабилизации БЛА; навигационные параметры БЛА в каждый момент времени (координаты и углы ориентации).

Заданные значения курса и высоты: 70° и 150 м.

Стартовые: 0° и 100 м.

Формат выходных данных

Программа на Arduino, реализующая автономный полет БЛА по заданным эшелону и курсу.

Критерии оценивания

- Максимальный балл за задачу — 10.
- Время на выход БЛА на заданный курс дается 4,6 с.
- Время на выход БЛА на заданную высоту дается 11,6 с.
- За каждые 0,5 с больше отведенного времени снимается 1 балл.
- Допустимое перерегулирование по углу курса — 4%.
- Допустимое перерегулирование по высоте — 1%.
- За каждые 0,5% процента превышения перерегулирования снимается 1 балл.
- За правильные ответы на теоретические вопросы (всего 2 вопроса) — 1 балл.

Решение

Вариант алгоритма решения задачи:

1. Рассчитать необходимый угол рысканья.
2. Выбрать структуру закона управления в канале крена для выхода на заданный курс по необходимому углу рысканья.
3. Выбрать структуру закона управления в канале тангажа для выхода на заданный эшелон.

4. Отладить разработанный алгоритм в симуляторе и проанализировать работу сформированных законов управления.

Пример программы-решения

Ниже представлено решение на языке C++.

```

1  #include "Tasks/Task2.h"
2
3  // Данная функция выполняется только один раз при запуске программы
4  // (при подаче питания на плату, перезагрузке, или нажатии кнопки reset)
5  void Task2_solution::init(HardwareSerial* a_Serial)
6  {
7      debug_serial = a_Serial;
8      // Примеры вывода в отладочную консоль
9      printtoDebugSerial("Hello team!");
10     printtoDebugSerial(String(3.1415926));
11 }
12
13 /*
14 ### УЧАСТНИКАМ ОЛИМПИАДЫ - Основная функция отправки пакета значений крена и
15 ↪ тангажа (град) ###
16 Данная функция осуществляет циклический обмен данных с симулятором с частотой 100
17 ↪ Гц.*/
18 SignalBody Task2_solution::Task8_in_the_loop(Skywalker2015PacketTelemetry
19 ↪ a_telemetry)
20 {
21     /* a_telemetry - структура данных, полученная с симулятора. Она содержит
22     ↪ текущие параметры БЛА
23     a_telemetry.L - координата X (север) [м]
24     a_telemetry.Z - координата Z (восток) [м]
25     a_telemetry.H - координата Y (высота) [м]
26     a_telemetry.Psi - угол рысканья [град]
27     a_telemetry.Gam - Угол крена [град]
28     a_telemetry.Tan - Угол тангажа [град]
29     a_telemetry.V - Скорость полета БЛА [м/с]
30     a_telemetry.Vx1 - Продольная скорость [м/с]
31     a_telemetry.Vz1 - Поперечная скорость [м/с]
32     a_telemetry.Vy1 - Вертикальная скорость [м/с]
33     a_telemetry.wx - Угловая скорость вокруг продольной оси [1/с]
34     a_telemetry.wy - Угловая скорость вокруг вертикальной оси [1/с]
35     a_telemetry.wz - Угловая скорость вокруг поперечной оси [1/с]
36     */
37
38     // РЕКОМЕНДУЕМЫЙ АЛГОРИТМ РЕШЕНИЯ ЗАДАЧИ 2
39
40     //
41     SignalBody _ans; // Структура которая отсылается на симулятор
42
43     float target_psi = -target_values.Yaw;
44     // На основе текущего значения угла рысканья и заданного угла рысканья
45     ↪ определяем необходимый угол крена
46     _ans.Gamma_direct = GammaReg(a_telemetry.Psi, target_psi);
47     // На основе текущей высоты БЛА, заданной высоты и вертикальной скорости
48     ↪ определяем необходимый угол тангажа
49     _ans.Tang_direct = HeightReg(a_telemetry.H, a_telemetry.Vy1,
50     ↪ target_values.Height);
51     // Отправляем команды на симулятор
52     // БЛА будет пытаться выдерживать заданные углы крена и тангажа

```

```

46     return _ans;
47 }
48
49 float roll_err_last = 0;
50 float Task2_solution::GammaReg(float Psi, float Psi_dir)
51 {
52     float Kpsi = 1.6;
53     float roll_err = AngDefines(Psi - Psi_dir);
54     float gamma_cmd = 2.0 * roll_err + 35 * (roll_err - roll_err_last);
55     roll_err_last = roll_err;
56     gamma_cmd = constrain(gamma_cmd, -30, 30);
57     return gamma_cmd;
58 }
59
60 float Task2_solution::HeightReg(float Yg, float Vy, float Hz)
61 {
62     float K_H = 0.56;
63     float K_vy_1 = 7.7591;
64     float K_vy_2 = 9.8941;
65     float kh1 = constrain(K_H * (Hz - Yg), -7.5, 7.5);
66     float _Pitch_direct = constrain((kh1 * K_vy_1 - Vy * K_vy_2), -20, 20);
67     return _Pitch_direct;
68 }

```

Материалы для подготовки

- Советы по настройке ПИД-регулятора: <https://realpars.com/pid-tuning/>.

Задача VI.2.5.3.

Для обеспечения управляемого полета БЛА самолетного типа оснащен множеством систем (органов) управления:

- руль высоты для разворота по тангажу;
- руль поворота для осуществления разворота по курсу;
- элероны для разворота по углу крена;
- двигатель для изменения скорости полета.

Для эффективного управления полетом требуется одновременное управление всеми этими системами.

Условие

Ознакомьтесь с органами управления летательного аппарата: двигателем, элеронами, рулем высоты и рулем направления. Напишите программу на esp32 (фреймворк Arduino), реализующую режим стабилизации по двум каналам (тангаж, крен).

Требуется реализовать:

- автоматическое отклонение органов управления для компенсации угла тангажа БВС;
- автоматическое отклонение органов управления для компенсации угла крена БВС;
- управление заданными положениями органов управления.

Формат выходных данных

Программа на esp32 (фреймворк Arduino), реализующая режим стабилизации БВС в горизонтальном полете по каналам крена и тангажа, с возможностью прямого управления.

Критерии оценивания

Факт достижения БВС заданного режима полета, а также реализацию функции прямого контроля органами управления оценивается организаторами на симуляторе и на полунатурном стенде:

- За успешную реализацию режима для каждого канала начисляется 4 балла.
- За успешную реализацию прямого управления органами управления БВС (руль высоты, руль направления, элероны) начисляется 2 балла.
- Правильный ответ на вопрос по теории прибавляет 1 балл (всего 2 вопроса).

Решение

Вариант алгоритма решения задачи:

1. Написать функцию прямого управления положением органов управления, отладить при помощи симулятора.
2. Показать организатору, применить на стенде ПНМ.
3. Написать функцию стабилизации полета по каналу тангажа и крена, отладить при помощи симулятора.
4. Показать организатору, применить на стенде ПНМ.

Пример программы-решения

Ниже представлено решение на языке C++.

```

1  #include "Tasks/Task3.h"
2  /***** Участникам олимпиады *****/
3  /*
4  Наименование функции:      Task3_in_the_loop()
5  Назначение функции:  Данная функция используется для обмена данными в режиме ТКПА
   ↳ симулятора
6  Входные данные приходящие с симулятора:
7
8  a_Elevator_zad - заданное значение руля высоты, град.
9  a_Rudder_zad - заданное значение руля направления, град.
10 a_Elerons_zad - заданное значения элеронов, град.
11 a_Thrust_zad - заданное значения тяги двигателя, Н.
12
13 a_Pitch_deg - текущее значение угла тангажа, град
14 a_Gamma_deg - текущее значение угла крена, град
15
16 Выходные данные идущие на симулятор:
17
18 Структура Task2_PWM содержащая следующие поля данных:
19
20 Elevator_PWM - RC сигнал, руль высоты

```

```

21 Rudder_PWM - RC сигнал, руль направления
22 Elerons_PWM - RC сигнал, элероны
23 Engine_RPM - RC сигнал, двигатель
24
25 Примечание: delay() в данной функции вызывать бесполезно, так как обмен с
   ↪ симулятором синхронный.
26
27 define SIM - включить когда отлаживаете ТОЛЬКО на симуляторе, когда переходите на
   ↪ стенд то необходимо закомментировать
28
29 */
30 // Закомментировать только когда отлаживаете на стенде.
31 #define SIM
32
33 Task3_PWM Task3_Solution::Task3_in_the_loop(const float& a_Elevator_zad, const
   ↪ float& a_Rudder_zad, const float& a_Elerons_zad, const float&
   ↪ a_Thrust_zad, const float &a_Pitch_deg, const float &a_Gamma_deg)
34 {
35     // Структура которая содержит заданный RC сигнал
36     Task3_PWM _ans;
37     // Текущие углы тангажа и крена с самолета (работают только при подключении к
   ↪ стенду)
38     double IMS_Pitch_deg = mav_orient.Pitch;    // Угол тангажа, градусы
39     double IMS_Roll_deg = mav_orient.Roll;      // Угол крена, градусы
40     // Текущие углы тангажа и крена поступающие с симулятора
41     double SIM_Pitch_deg = a_Pitch_deg;
42     double SIM_Roll_deg = a_Gamma_deg;
43
44     // Функция прямого контроля положения органов управления
45     _ans = full_control_mode(a_Elevator_zad, a_Rudder_zad, a_Elerons_zad);
46
47     // Функция стабилизации, в зависимости от текущих углов приходящих с
   ↪ симулятора
48     // _ans = stab_mode(SIM_Pitch_deg, SIM_Roll_deg);
49
50
51     #ifndef SIM
52
53     // Функция стабилизации, в зависимости от текущих углов приходящих с стенда
54     _ans = stab_mode(IMS_Pitch_deg, IMS_Roll_deg);
55
56     // Подаем структуру сигналов на Ardupilot (Участникам не трогать)
57     if ((millis() - t_last_control) > control_interval)
58     {
59         mav_rc_override(_ans);
60     }
61     #endif
62
63     // Возвращаем RC сигнал на симулятор
64     return _ans;
65 }
66 // Функция прямого контроля положения органов управления
67 Task3_PWM Task3_Solution::full_control_mode(const float& a_Elevator_zad, const
   ↪ float& a_Rudder_zad, const float& a_Elerons_zad)
68 {
69     // Структура которая содержит заданный RC сигнал
70     Task3_PWM _ans;
71     _ans.Elerons_PWM = map(a_Elerons_zad, ELERONS_DEG_MIN, ELERONS_DEG_MAX,
   ↪ ALERON_PWM_MIN, ALERON_PWM_MAX);
72     _ans.Elevator_PWM = map(a_Elevator_zad, ELEVATOR_DEG_MIN, ELEVATOR_DEG_MAX,
   ↪ ELEVATOR_PWM_MIN, ELEVATOR_PWM_MAX);

```

```

73     _ans.Rudder_PWM = map(a_Rudder_zad, RUDDER_DEG_MIN, RUDDER_DEG_MAX,
74     ↪ RUDDER_PWM_MIN, RUDDER_PWM_MAX);
75     return _ans;
76 }
77 // Функция стабилизации, в зависимости от текущих углов
78 Task3_PWM Task3_Solution::stab_mode(double Pitch, double Roll)
79 {
80     // Структура которая содержит заданный RC сигнал
81     Task3_PWM _ans;
82
83     double Roll_zad = -1.3*Roll;
84     double Pitch_zad = 1.2*Pitch;
85     double Head_zad = 0.1*Roll;
86
87     // эталонка
88     _ans.Elerons_PWM = map(Roll_zad, ELERONS_DEG_MIN, ELERONS_DEG_MAX,
89     ↪ ALERON_PWM_MIN, ALERON_PWM_MAX);
90     _ans.Rudder_PWM = map(Head_zad, RUDDER_DEG_MIN, RUDDER_DEG_MAX,
91     ↪ RUDDER_PWM_MIN, RUDDER_PWM_MAX);
92     _ans.Elevator_PWM = map(Pitch_zad, ELEVATOR_DEG_MIN, ELEVATOR_DEG_MAX,
93     ↪ ELEVATOR_PWM_MIN, ELEVATOR_PWM_MAX);
94
95     return _ans;
96 }
97
98 void Task3_Solution::init(HardwareSerial * a_mav_serial)
99 {
100     _mav_serial = a_mav_serial;
101     // Запускаем обмен с Matek
102     init_base(_mav_serial);
103 }
104
105 void Task3_Solution::init_via_debug(HardwareSerial * a_mav_serial, HardwareSerial *
106 ↪ a_debug)
107 {
108     _mav_serial = a_mav_serial;
109     _debug = a_debug;
110     #ifndef SIM
111     // Запускаем обмен с Ardupilot
112     _debug->println("Matek initialization start...");
113     init_base(_mav_serial);
114     #endif
115 }
116
117 void Task3_Solution::mav_rc_override(Task3_PWM a_pwm)
118 {
119     mavlink_message_t msg;
120     uint8_t buf[MAVLINK_MAX_PACKET_LEN];
121
122     // mavlink_msg_rc_channels_override_pack(0xFF, 0xBE, &msg, 1, 1, roll, pitch,
123     ↪ throttle, yaw, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0);
124     mavlink_msg_rc_channels_override_pack(0xFF, 0xBE, &msg, (uint8_t)1,
125     ↪ (uint8_t)1, (uint16_t)a_pwm.Elerons_PWM, (uint16_t)a_pwm.Elevator_PWM,
126     ↪ (uint16_t)(1100), (uint16_t)a_pwm.Rudder_PWM, UINT16_MAX, UINT16_MAX,
127     ↪ UINT16_MAX, UINT16_MAX, UINT16_MAX, UINT16_MAX, UINT16_MAX, UINT16_MAX,
128     ↪ UINT16_MAX, UINT16_MAX, UINT16_MAX, UINT16_MAX, UINT16_MAX, UINT16_MAX);
129     uint16_t len = mavlink_msg_to_send_buffer(buf, &msg);
130
131     _mav_serial->write(buf, len);
132     t_last_control = millis();
133 }

```

Материалы для подготовки

- ШИМ-сигнал в Arduino: <https://alexgyver.ru/lessons/pwm-signal/>.

Задача VI.2.5.4.

Одной из базовых функций, реализуемых БЛА, является автономный полет по заданной траектории. Для осуществления такого полета разрабатываются специальные системы автоматического управления, способные регулировать все параметры навигации БЛА в зависимости от его положения относительно заданной траектории.

Условие

Разработайте программу, реализующую автоматическое управление углом крена и тангажа БЛА для осуществления его движения по заданной траектории. Траектория движения формируется участниками команды произвольным образом с целью пролета БЛА по расположенным в симуляторе путевым точкам.

Формат входных данных

Координаты путевых точек, система автоматического управления БЛА (кроме регуляторов в каналах крена и тангажа); навигационные параметры БЛА в каждый момент времени (координаты и углы ориентации).

Формат выходных данных

Программа на Arduino, реализующая автономный полет БЛА по заданным путевым точкам.

Критерии оценивания

В симуляторе расположены 20 путевых точек в виде сфер диаметром 15 м. Оценивается количество пройденных БЛА путевых точек.

- За каждую пройденную путевую точку начисляется 0,5 балла.
- Время полета БЛА не должно превышать 6 мин.
- Итоговый результат всегда округляется в большую сторону.
- Правильный ответ на вопрос по теории прибавляет 1 балл (всего 2 вопроса).

Решение

Вариант алгоритма решения задачи:

1. Выбрать структуру законов управления в каналах крена и тангажа.
2. Определить необходимые значения угла курса и высоты для достижения следующей путевой точки.
3. Выбрать траекторию движения БЛА (последовательность путевых точек) с учетом заданного ограничения по времени.

4. Отработать разработанный алгоритм на симуляторе.
 5. Проанализировать полученный результат и доработать алгоритмы управления.
- Пункты 4 и 5 повторять необходимое количество раз.

Пример программы-решения

Ниже представлено решение на языке C++.

```

1  #include "Tasks/Task4.h"
2
3  // Данная функция выполняется только один раз при запуске программы
4  // (при подаче питания на плату, перезагрузке, или нажатии кнопки reset)
5  void Task4_solution::init(HardwareSerial* a_Serial)
6  {
7      debug_serial = a_Serial;
8      // Примеры вывода в отладочную консоль
9      printtoDebugSerial("Hello team!");
10     printtoDebugSerial(String(3.1415926));
11
12     // Путьевые точки в произвольном порядке
13     _PointsArray[0] = FlyPoints(0, 250, 70);
14
15     _PointsArray[1] = FlyPoints(0, 500, 90);
16     _PointsArray[2] = FlyPoints(250, 500, 70);
17     _PointsArray[3] = FlyPoints(500, 500, 70);
18     _PointsArray[4] = FlyPoints(250, 250, 90);
19     _PointsArray[5] = FlyPoints(250, 250, 80);
20     _PointsArray[6] = FlyPoints(500, 0, 70);
21     _PointsArray[7] = FlyPoints(500, -250, 90);
22     _PointsArray[8] = FlyPoints(250, -250, 70);
23
24     _PointsArray[9] = FlyPoints(0, -250, 90);
25     _PointsArray[10] = FlyPoints(0, -250, 80);
26     _PointsArray[11] = FlyPoints(0, -500, 70);
27     _PointsArray[12] = FlyPoints(-250, -500, 80);
28     _PointsArray[13] = FlyPoints(-250, -250, 70);
29     _PointsArray[14] = FlyPoints(-500, 0, 90);
30     _PointsArray[15] = FlyPoints(-500, 250, 80);
31     _PointsArray[16] = FlyPoints(-250, 500, 80);
32     _PointsArray[17] = FlyPoints(-250, 250, 80);
33     _PointsArray[18] = FlyPoints(0, 0, 90);
34     _PointsArray[19] = FlyPoints(0, 0, 80);
35 }
36
37 /*
38 ### УЧАСТНИКАМ ОЛИМПИАДЫ - Основная функция отправки пакета значений крена и
39 ↪ тангажа (град) ###
40 Данная функция осуществляет циклический обмен данных с симулятором с частотой 100
41 ↪ Гц.
42 */
43 SignalBody Task4_solution::Task8_in_the_loop(Skywalker2015PacketTelemetry
44 ↪ a_telemetry)
45 {
46     /* a_telemetry - структура данных, полученная с симулятора. Она содержит
47     ↪ текущие параметры БЛА
48     a_telemetry.L - координата X (север) [м]
49     a_telemetry.Z - координата Z (восток) [м]
50     a_telemetry.H - координата Y (высота) [м]

```

```

47     a_telemetry.Psi - угол рысканья [град]
48     a_telemetry.Gam - Угол крена [град]
49     a_telemetry.Tan - Угол тангажа [град]
50     a_telemetry.V - Скорость полета БЛА [м/с]
51     a_telemetry.Vx1 - Продольная скорость [м/с]
52     a_telemetry.Vz1 - Поперечная скорость [м/с]
53     a_telemetry.Vy1 - Вертикальная скорость [м/с]
54     a_telemetry.wх - Угловая скорость вокруг продольной оси [1/с]
55     a_telemetry.wу - Угловая скорость вокруг вертикальной оси [1/с]
56     a_telemetry.wz - Угловая скорость вокруг поперечной оси [1/с]
57     */
58
59     // РЕКОМЕНДУЕМЫЙ АЛГОРИТМ РЕШЕНИЯ ЗАДАЧИ 4
60     //
61     SignalBody_ans; // Структура которая отправляется на симулятор
62     // Проверить близость БЛА к текущей путевой точке. Изменить путевую точку при
        ↪ необходимости
63     _Point_Index = GetNowPointIndex(a_telemetry.L, a_telemetry.Z, a_telemetry.H);
64     // Через координаты путевой точки и текущего местоположения БЛА найти
        ↪ направление на путевую точку (пеленг)
65     _ans.Gamma_direct = PointsFlyGam(_Point_Index, a_telemetry.L, a_telemetry.Z,
        ↪ a_telemetry.Psi);
66     // Рассчитать необходимое изменение угла курса
67     // Рассчитать требуемый угол крена (для регулирования курса)
68     // На основе текущей высоты БЛА и заданной высоты путевой точки рассчитать
        ↪ необходимое изменение высоты
69     _ans.Tang_direct = PointsFlyTan(a_telemetry.H, a_telemetry.Vy1,
        ↪ _Point_Index);
70     // Рассчитать требуемый угол тангажа (для регулирования высоты)
71
72     //_ans.Gamma_direct = 0; // заданный угол крена
73     //_ans.Tang_direct = 0; // заданный угол тангажа
74
75     // Отправляем команды на симулятор
76     // БЛА будет пытаться выдерживать заданные углы крена и тангажа
77     return _ans;
78 }
79
80 float roll_err_last = 0;
81 float Task4_solution::GammaReg(float Psi, float Psi_dir)
82 {
83     float Kpsi = 1.6;
84     float roll_err = AngDefines(Psi - Psi_dir);
85     float gamma_cmd = 2.0 * roll_err + 35 * (roll_err - roll_err_last);
86     roll_err_last = roll_err;
87     gamma_cmd = constrain(gamma_cmd, -30, 30);
88     return gamma_cmd;
89 }
90
91 float Task4_solution::HeightReg(float Yg, float Vy, float Hz)
92 {
93     float K_H = 0.56;
94     float K_vy_1 = 7.7591;
95     float K_vy_2 = 9.8941;
96     float kh1 = constrain(K_H * (Hz - Yg), -7.5, 7.5);
97     float _Pitch_direct = constrain((kh1 * K_vy_1 - Vy * K_vy_2), -20, 20);
98     return _Pitch_direct;
99 }
100
101 float Task4_solution::ToPointXY(float _Xt, float _Yt, float a_Xg, float a_Zg,
        ↪ float Psi)

```

```

102 {
103     float Psi_cmd = -atan2(a_Zg - _Yt, a_Xg - _Xt);
104     Psi_cmd *= 57.3;
105     return GammaReg(Psi, Psi_cmd);
106 }
107
108 float Task4_solution::PointsFlyGam(const size_t& a_Point_index, float _Xt, float
↳ _Yt, float Psi)
109 {
110     float Xg_cmd = _PointsArray[_Point_Index].North;
111     float Zg_cmd = _PointsArray[_Point_Index].East;
112     return ToPointXY(_Xt, _Yt, Xg_cmd, Zg_cmd, Psi);
113 }
114
115 float Task4_solution::PointsFlyTan(float H, float Vy, const size_t& a_Point_index)
116 {
117     float H_z = _PointsArray[_Point_Index].Height;
118     return HeightReg(H, Vy, H_z);
119 }
120
121 size_t Task4_solution::GetNowPointIndex(float X, float Z, float H)
122 {
123     size_t max_index = sizeof(_PointsArray) / sizeof(_PointsArray[0]);
124
125     float Xg_cmd = _PointsArray[_Point_Index].North;
126     float Zg_cmd = _PointsArray[_Point_Index].East;
127     float Yg_cmd = _PointsArray[_Point_Index].Height;
128
129     float delX = Xg_cmd - X;
130     float delY = Zg_cmd - Z;
131     float delZ = Yg_cmd - H;
132
133     float Size = sqrtf(pow(delX, 2) + pow(delY, 2) + pow(delZ, 2));
134
135     // Calculate crosstrack error
136     float x_og = 0.0;
137     float z_og = 0.0;
138     if (_Point_Index > 0)
139     {
140         x_og = _PointsArray[_Point_Index - 1].North;
141         z_og = _PointsArray[_Point_Index - 1].East;
142     }
143     else
144     {
145         x_og = _PointsArray[max_index - 1].North;
146         z_og = _PointsArray[max_index - 1].East;
147     }
148     float relx = X - x_og;
149     float rely = Z - z_og;
150     float delx = Xg_cmd - x_og;
151     float dely = Zg_cmd - z_og;
152     float U = (relx * delx + rely * dely) / (delx * delx + dely * dely);
153     float cte = (rely * delx - relx * dely) / (delx * delx + dely * dely);
154
155     if (Size < 7.0)
156     {
157         _Point_Index++;
158     }
159
160     if (_Point_Index >= max_index)

```

```
161     {  
162         _Point_Index = 0;  
163     }  
164     return _Point_Index;  
165 }
```

Задача VI.2.5.5.

Чтобы успешно доставить груз в назначенное место, важно не только проходить через контрольные точки, но и точно идентифицировать зону доставки груза на фотоснимке. В этом помогает камера, установленная на БЛА, которая обеспечивает съемку местности под аппаратом.

Условие

Разработайте программу, программу, позволяющую обнаружить наличие зоны доставки груза на фотографии и определите координаты точки в этой зоне.

Формат входных данных

Изображение, на котором есть зона доставки груза, два изображения на которых нет зоны доставки груза, высота полета и координаты БЛА в момент съемки изображений; углов отклонения по крену и тангажу в момент съемки нет.

Формат выходных данных

Программа на языке Python, выводящая информацию о наличии на изображениях зоны доставки и, в случае обнаружения, координаты точки внутри этой зоны.

Критерии оценивания

- За правильное обнаружение зоны доставки груза начисляется 4 балла.
- За правильное определение точки доставки начисляется 6 баллов.
- За ошибку в определении координат точки доставки в пределах 30 м каждые 5 метров снимается по баллу. В случае, если ошибка больше 30 м точка считается некорректно обнаруженной.
- За ошибочное обнаружение на изображениях где нет зоны доставки —5 баллов.
- За вопросы по теории можно получить еще по 1 баллу за вопрос (всего два).

Решение

Вариант алгоритма решения задачи:

1. Разработать алгоритм поиска здания.
2. Разработать алгоритм определения положения автомобиля на кадре.
3. Разработать алгоритм расчета координат точки по ее координатам в кадре.
4. Отладить разработанные алгоритмы на изображениях.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```

1 import cv2
2 import matplotlib.pyplot as plt
3 import numpy as np
4
5 class Solution:
6     f = 6.1 * 1e-3 # Фокусное расстояние
7     ph = 7.8 * 1e-6 # Высота пикселя
8     pw = 9.2 * 1e-6 # Ширина пикселя
9     resolution = (720, 1280) # Размер изображения в пикселях
10    camera_zxy = (200, 300, 100) # Координаты места съёмки кадра Z, X, Y
11
12    def check_building(self, hsv):
13        building_mask_lower = np.array([0, 0, 190])
14        building_mask_upper = np.array([180, 50, 255])
15        building_mask = cv2.inRange(hsv, building_mask_lower, building_mask_upper)
16        contours, _ = cv2.findContours(building_mask, cv2.RETR_EXTERNAL,
17        ↪ cv2.CHAIN_APPROX_SIMPLE)
18        if contours:
19            largest_contour = max(contours, key=cv2.contourArea)
20            x, z, w, h = cv2.boundingRect(largest_contour)
21            rectangle_area = w * h
22            if rectangle_area > 30000 and w/h > 0.5 and w/h < 1.5:
23                return True
24        return False
25
26    def find_delivery_point(self, fname):
27        image = cv2.imread(fname)
28        image_h, image_w = self.resolution
29
30        lower_green = np.array([40, 40, 40])
31        upper_green = np.array([100, 255, 255])
32
33        hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
34
35        if self.check_building(hsv):
36            mask = cv2.inRange(hsv, lower_green, upper_green)
37            car_contours, _ = cv2.findContours(mask, cv2.RETR_TREE,
38            ↪ cv2.CHAIN_APPROX_SIMPLE)
39            largest_contour = max(car_contours, key=cv2.contourArea)
40            z, x, w, h = cv2.boundingRect(largest_contour)
41            if (w*h < 1500 and w*h > 500 and w/h > 0.7 and w/h < 10):
42
43                z = (z+w//2) - image_w//2
44                x = ((image_h//2 - (x+h//2)))
45
46                result_z, result_x = self.point_coordinates(self.camera_zxy, (z,
47                ↪ x))
48                return result_z+3, result_x
49        return None
50
51    def point_coordinates(self, zxy_camera, zx_point):
52        z_camera, x_camera, y_camera = zxy_camera
53        z_point, x_point = zx_point
54
55        delta_w_m = z_point * self.pw
56        delta_h_m = x_point * self.ph

```

```
54
55     z_point = z_camera + (delta_w_m * y_camera) / self.f
56     x_point = x_camera + (delta_h_m * y_camera) / self.f
57
58     return z_point, x_point
59
60 if __name__=='__main__':
61     image_paths = ['img1.jpg', 'img2.jpg', 'img3.jpg']
62
63     solver = Solution()
64     for image_path in image_paths:
65         print(solver.find_delivery_point(image_path))
```

Материалы для подготовки

- Библиотека OpenCV <https://stepik.org/course/116539/promo>.

Задача VI.2.5.6.

Одной из перспективных областей применения БЛА является оперативная доставка медицинских грузов в труднодоступные места. Это направление развития беспилотных систем особенно актуально для регионов России с низкой транспортной доступностью.

Условие

Разработайте программу на языке Python, реализующую автоматическое траекторное управление полетом БЛА и определение координат точки доставки медицинского груза посредством обработки изображений с бортовой камеры БЛА в режиме реального времени.

Формат входных данных

Симулятор полета БЛА; навигационные параметры БЛА в текущий момент времени; видеопоток данных с бортовой камеры БЛА.

Формат выходных данных

Программа на Python, реализующая расчет координат местоположения точки доставки груза с использованием алгоритмов технического зрения; обработанное изображение с обозначением зоны доставки груза.

Критерии оценивания

Оценивается точность определения координат точки доставки медицинского груза. Оценка осуществляется путем расчета расстояния от точки доставки до координат местоположения ориентира:

- Если расстояние от точки доставки до ориентира меньше 2 м — за решение задачи начисляется 0 баллов.

- Если расстояние лежит в промежутке [2, 7] — за решение задачи начисляется максимальная оценка — 10 баллов.
- За каждые дополнительные 2 м расстояния снимается 1 балл.
- Правильный ответ на вопрос по теории прибавляет 1 балл (всего 2 вопроса).

Решение

Вариант алгоритма решения задачи:

1. Сформировать траекторию полета БЛА; написать код алгоритма траекторного управления для реализации этой траектории.
2. Разработать алгоритм обработки изображений с бортовой камеры БЛА.
3. Отладить алгоритм для исключения ложных срабатываний.
4. Разработать алгоритм определения координат точки доставки груза.
5. Отработать разработанные алгоритмы на симуляторе.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```

1  # -*- coding: utf-8 -*-
2  import socket
3  from threading import Thread
4  import time
5  from math import atan2, cos, sin, sqrt, pi, atan
6  import cv2
7  import numpy as np
8  import uav_state
9  import serial
10
11 import subprocess
12
13 # Последние полученные данные телеметрии
14 last_telemetry = uav_state.uav_state()
15 running = True
16 capture = cv2.VideoCapture()
17
18 ser = serial.Serial(
19     port='COM6',
20     baudrate = 115200,
21     parity = serial.PARITY_NONE,
22     stopbits = serial.STOPBITS_ONE,
23     bytesize = serial.EIGHTBITS,
24     timeout = 0.1 )
25
26 # UDP сервер для параллельного получения данных телеметрии
27 def udp_server_thread():
28     host = '127.0.0.1'
29     port = 63504
30     addr = (host, port)
31     server = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
32     server.bind(addr)
33
34     while running:
35         d = server.recvfrom(1024)

```

```

36     received = d[0]
37     addr = d[1]
38
39     received = received.decode("utf-8")
40     received = received.replace("\t\r\n", "")
41     #print("---")
42     #print(received)
43     arr = [float(i) for i in received.split('\t')]
44     if len(arr) == 18:
45         last_telemetry.update(arr)
46     server.close()
47
48 def sendNewWaypoint(newx, newz, newh):
49     ser.write("wpnt {:.2f} {:.2f} {:.2f} \n".format(newx, newz, newh).encode())
50     print("wpnt {:.2f} {:.2f} {:.2f}".format(newx, newz, newh))
51
52 def sendDropCargo():
53     ser.write("drop \n".encode())
54     print("cargo drop attempt!")
55
56 # УЧАСТНИКАМ НТО - ЗАДАЧИ 6,7,8 -----
57 def find_surroundings(image, center, radius):
58     dots = []
59     h, w = image.shape
60     for row in range(center[1] - radius, center[1] + radius):
61         if row < h and row >= 0:
62             dots.append(image[row][center[0]])
63     for col in range(center[0] - radius, center[0] + radius):
64         if col < w and col >= 0:
65             dots.append(image[center[1]][col])
66
67     avg_val = sum(dots) / len(dots)
68     return avg_val
69
70 # Поток обработки данных с камеры
71 def camera_processing_thread():
72     # Параметры камеры
73     F = 6.1 * 1e-3 # Фокусное расстояние
74     dv = 7.8 * 1e-6 # высота пикселя
75     dh = 9.2 * 1e-6 # ширина пикселя
76     xadj = 2
77     zadj = -8
78     cap = cv2.VideoCapture("rtp-forwarder.sdp")
79     #cap = cv2.VideoCapture("red3.png")
80     cap.set(cv2.CAP_PROP_BUFFERSIZE, 0)
81
82
83     gray_lower = np.array([0, 00, 80])
84     gray_upper = np.array([255, 170, 170])
85
86     red_lower_v = np.array([150, 90, 20])
87     red_upper_v = np.array([250, 250, 245])
88     print("ready to search")
89
90     sendNewWaypoint(400, 600, 80)
91     # Основной цикл программы
92     while running:
93         # Получение изображения с бортовой камеры
94         ret, frame = cap.read()
95         if ret:

```

```

96     tele = last_telemetry
97     X = tele.x
98     Z = tele.z - 8
99     H = tele.h
100    Heading = tele.yaw * pi / 180
101
102    height, width, channels = frame.shape
103    cv2.normalize(frame, frame, 0, 255, cv2.NORM_MINMAX)
104    frame = cv2.GaussianBlur(frame, (7, 7), 0)
105    hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
106    gray_mask = cv2.inRange(hsv, gray_lower, gray_upper)
107    red2_mask = cv2.inRange(hsv, red_lower_v, red_upper_v)
108    full_mask = red2_mask #+ red3_mask
109    contours, hierarchy = cv2.findContours(full_mask, cv2.RETR_LIST,
110    ↪ cv2.CHAIN_APPROX_SIMPLE)
111    maxlen = 0
112    if len(contours) > 0:
113        truecont = contours[0]
114        for icontour in contours:
115            if len(icontour) > maxlen:
116                maxlen = len(icontour)
117                truecont = icontour
118            ## print(f"max green contour: {maxlen}")
119            cv2.drawContours(frame, truecont, -1, (0,0,255), 2)
120            sum_x = 0
121            sum_y = 0
122            for i in range(0, len(truecont)):
123                sum_x += truecont[i][0][0]
124                sum_y += truecont[i][0][1]
125            avg_x = int(round(sum_x / len(truecont)))
126            avg_y = int(round(sum_y / len(truecont)))
127            park = find_surroundings(gray_mask, (avg_x, avg_y), 40)
128            print(f"parking lot value: {park:.1f}")
129            if park < 25:
130                ##print("Found the car on the parking lot : {} hor, {} ver
131                ↪ pixels from the corner".format(avg_x, avg_y))
132                cv2.ellipse(frame, (avg_x, avg_y), (5, 5), 0, 0, 360, (255,
133                ↪ 255, 255))
134                dxp = avg_x - width / 2
135                dyp = height/2 - avg_y
136                dxx = H / F * dh
137                dxy = H / F * dv
138                ## print(f"Pixel size in meters: {dxx}")
139                shift_x = dxp * dxx
140                shift_y = dyp * dxy
141                ## print("Contour center location: {} hor, {} ver pixels from
142                ↪ the center".format(dxp, dyp))
143                ## print("Contour center location: {} hor, {} ver meters from
144                ↪ the center".format(shift_x, shift_y))
145                geo_x = shift_y * cos(Heading) - shift_x * sin(Heading) + X
146                geo_y = shift_y * sin(Heading) + shift_x * cos(Heading) + Z
147                ##print("UAV coords X: {:.2f} Z: {:.2f} Heading: {}".format(X, Z,
148                ↪ Heading))
149                print("Car coords X: {:.2f} Z: {:.2f}".format(geo_x, geo_y))
150                print("Delivery coords X: {:.2f} Z: {:.2f}".format(geo_x + xadj,
151                ↪ geo_y + zadj))
152
153    result = cv2.putText(frame, f"{(geo_x + xadj):.1f} {(geo_y +
154    ↪ zadj):.1f}", (int(width/2), int(height/2)), \
155    ↪ cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 0,
156    ↪ 0), 2, cv2.LINE_AA)

```

```

148         cv2.imwrite("result.png", result)
149         #cv2.waitKey()
150         return
151
152     cv2.imshow("computer vision", frame)
153     cv2.waitKey(3)
154     else:
155         time.sleep(0.05)
156         print("no signal")
157
158     # Основная функция для управления потоками
159     if __name__ == '__main__':
160         # Open RTP livestream through golang script
161         subprocess.Popen(["go", "run", "./main.go"])
162         time.sleep(5)
163         capture_thr = Thread(target=udp_server_thread, args=[])
164         capture_thr.daemon = True
165         capture_thr.start()
166         process_thr = Thread(target=camera_processing_thread, args=[])
167         process_thr.daemon = True
168         process_thr.start()
169
170     try:
171         while running:
172             time.sleep(1)
173     except KeyboardInterrupt:
174         print('\ninterrupted!')
```

Материалы для подготовки

- Библиотека OpenCV <https://stepik.org/course/116539/promo>.

Задача VI.2.5.7.

Одной из перспективных областей применения БЛА является оперативная доставка медицинских грузов в труднодоступные места. Это направление развития беспилотных систем особенно актуально для регионов России с низкой транспортной доступностью.

Условие

Разработайте программу на языке Python, реализующую автоматическое траекторное управление полетом БЛА и автономную доставку медицинского груза с использованием системы технического зрения.

Формат входных данных

Симулятор полета БЛА; навигационные параметры БЛА в текущий момент времени; видеопоток данных с бортовой камеры БЛА.

Формат выходных данных

Программа на Python, реализующая расчет координат местоположения точки доставки груза с использованием алгоритмов технического зрения и автоматическую доставку груза.

Критерии оценивания

Оценивается точность доставки медицинского груза. Оценка осуществляется путем расчета расстояния от фактической точки доставки до координат местоположения ориентира:

- Если расстояние от точки доставки до ориентира меньше 2 м, то за решение задачи начисляется 0 баллов.
- Если расстояние лежит в промежутке [2, 7] — за решение задачи начисляется максимальная оценка — 12 баллов.
- За каждые дополнительные 2 м расстояния снимается 1 балл.
- Правильный ответ на вопрос по теории прибавляет 1 балл (всего 2 вопроса).

Решение

Вариант алгоритма решения задачи:

1. Сформировать траекторию полета БЛА; написать код алгоритма траекторного управления для реализации этой траектории.
2. Оценить физику движения груза.
3. Разработать алгоритм сброса груза.
4. Разработать алгоритм определения координат точки доставки груза.
5. Отработать разработанные алгоритмы на симуляторе.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```

1  # -*- coding: utf-8 -*-
2  import socket
3  from threading import Thread
4  import time
5  from math import atan2, cos, sin, sqrt, pi, atan
6  import cv2
7  import numpy as np
8  import uav_state
9  import serial
10
11 import subprocess
12
13 # Последние полученные данные телеметрии
14 last_telemetry = uav_state.uav_state()
15 running = True
16 capture = cv2.VideoCapture()
17
18 ser = serial.Serial(
```

```

19     port='COM6',
20     baudrate = 115200,
21     parity = serial.PARITY_NONE,
22     stopbits = serial.STOPBITS_ONE,
23     bytesize = serial.EIGHTBITS,
24     timeout = 0.1 )
25
26 # UDP сервер для параллельного получения данных телеметрии
27 def udp_server_thread():
28     host = '127.0.0.1'
29     port = 63504
30     addr = (host, port)
31     server = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
32     server.bind(addr)
33
34     while running:
35         d = server.recvfrom(1024)
36         received = d[0]
37         addr = d[1]
38
39         received = received.decode("utf-8")
40         received = received.replace("\t\r\n", "")
41         #print("---")
42         #print(received)
43         arr = [float(i) for i in received.split('\t')]
44         if len(arr) == 18:
45             last_telemetry.update(arr)
46     server.close()
47
48 def sendNewWaypoint(newx, newz, newh):
49     ser.write("wpnt {:.2f} {:.2f} {:.2f} \n".format(newx, newz, newh).encode())
50     print("wpnt {:.2f} {:.2f} {:.2f}".format(newx, newz, newh))
51
52 def sendDropCargo():
53     ser.write("drop \n".encode())
54     print("cargo drop attempt!")
55
56 # УЧАСТНИКАМ НТО - ЗАДАЧИ 6,7,8 -----
57 def find_surroundings(image, center, radius):
58     dots = []
59     h, w = image.shape
60     for row in range(center[1] - radius, center[1] + radius):
61         if row < h and row >= 0:
62             dots.append(image[row][center[0]])
63     for col in range(center[0] - radius, center[0] + radius):
64         if col < w and col >= 0:
65             dots.append(image[center[1]][col])
66
67     avg_val = sum(dots) / len(dots)
68     return avg_val
69
70 # Поток обработки данных с камеры
71 def camera_processing_thread():
72     # Параметры камеры
73     F = 6.1 * 1e-3 # Фокусное расстояние
74     dv = 7.8 * 1e-6 # высота пикселя
75     dh = 9.2 * 1e-6 # ширина пикселя
76     xadj = 2
77     zadj = -8
78     cap = cv2.VideoCapture("rtp-forwarder.sdp")

```

```

79     #cap = cv2.VideoCapture("red3.png")
80     cap.set(cv2.CAP_PROP_BUFFERSIZE, 0)
81
82
83     gray_lower = np.array([0, 00, 80])
84     gray_upper = np.array([255, 170, 170])
85
86     red_lower_v = np.array([150, 90, 20])
87     red_upper_v = np.array([250, 250, 245])
88     print("ready to search")
89
90     sendNewWaypoint(400, 600, 80)
91     # Основной цикл программы
92     while running:
93         # Получение изображения с бортовой камеры
94         ret, frame = cap.read()
95         if ret:
96             tele = last_telemetry
97             X = tele.x
98             Z = tele.z - 8
99             H = tele.h
100            Heading = tele.yaw * pi / 180
101
102            height, width, channels = frame.shape
103            cv2.normalize(frame, frame, 0, 255, cv2.NORM_MINMAX)
104            frame = cv2.GaussianBlur(frame, (7, 7), 0)
105            hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
106            gray_mask = cv2.inRange(hsv, gray_lower, gray_upper)
107            red2_mask = cv2.inRange(hsv, red_lower_v, red_upper_v)
108            full_mask = red2_mask #+ red3_mask
109            contours, hierarchy = cv2.findContours(full_mask, cv2.RETR_LIST,
110            ↪ cv2.CHAIN_APPROX_SIMPLE)
111            maxlen = 0
112            if len(contours) > 0:
113                truecont = contours[0]
114                for icontour in contours:
115                    if len(icontour) > maxlen:
116                        maxlen = len(icontour)
117                        truecont = icontour
118                # # print(f"max green contour: {maxlen}")
119                cv2.drawContours(frame, truecont, -1, (0,0,255), 2)
120                sum_x = 0
121                sum_y = 0
122                for i in range(0, len(truecont)):
123                    sum_x += truecont[i][0][0]
124                    sum_y += truecont[i][0][1]
125                avg_x = int(round(sum_x / len(truecont)))
126                avg_y = int(round(sum_y / len(truecont)))
127                park = find_surroundings(gray_mask, (avg_x, avg_y), 40)
128                print(f"parking lot value: {park:.1f}")
129                if park < 25:
130                    #print("Found the car on the parking lot : {} hor, {} ver
131                    ↪ pixels from the corner".format(avg_x, avg_y))
132                    cv2.ellipse(frame, (avg_x, avg_y), (5, 5), 0, 0, 360, (255,
133                    ↪ 255, 255))
134                    dxp = avg_x - width / 2
135                    dyp = height/2 - avg_y
136                    dxx = H / F * dh
137                    dxy = H / F * dv
138                    # print(f"Pixel size in meters: {dxx}")

```

```

136         shift_x = dxp * dx
137         shift_y = dyp * dy
138         # print("Contour center location: {} hor, {} ver pixels from
↳ the center".format(dxp, dyp))
139         # print("Contour center location: {} hor, {} ver meters from
↳ the center".format(shift_x, shift_y))
140         geo_x = shift_y * cos(Heading) - shift_x * sin(Heading) + X
141         geo_y = shift_y * sin(Heading) + shift_x * cos(Heading) + Z
142         #print("UAV coords X:{:2f} Z:{:2f} Heading:{}".format(X, Z,
↳ Heading))
143         print("Car coords X:{:2f} Z:{:2f}".format(geo_x, geo_y))
144         print("Delivery coords X:{:2f} Z:{:2f}".format(geo_x + xadj,
↳ geo_y + zadj))
145
146         result = cv2.putText(frame, f"{{(geo_x + xadj):.1f}} {{(geo_y +
↳ zadj):.1f}}", (int(width/2), int(height/2)), \
147                             cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 0,
↳ 0), 2, cv2.LINE_AA)
148         cv2.imwrite("result.png", result)
149         #cv2.waitKey()
150         return
151
152         cv2.imshow("computer vision", frame)
153         cv2.waitKey(3)
154     else:
155         time.sleep(0.05)
156         print("no signal")
157
158     # Основная функция для управления потоками
159     if __name__ == '__main__':
160         # Open RTP livestream through golang script
161         subprocess.Popen(["go", "run", "./main.go"])
162         time.sleep(5)
163         capture_thr = Thread(target=udp_server_thread, args=[])
164         capture_thr.daemon = True
165         capture_thr.start()
166         process_thr = Thread(target=camera_processing_thread, args=[])
167         process_thr.daemon = True
168         process_thr.start()
169
170     try:
171         while running:
172             time.sleep(1)
173     except KeyboardInterrupt:
174         print('\ninterrupted!')
```

Материалы для подготовки

- Библиотека OpenCV <https://stepik.org/course/116539/promo>.

Задача VI.2.5.8.

Своевременная доставка медицинских грузов в труднодоступные места требует решения целого ряда задач. Одна из этих задач – логистическая: как и куда нужно лететь, где сбросить груз, в каком порядке осуществлять доставку.

Условие

Разработайте программу на языке Python, реализующую автоматическое траекторное управление полетом БЛА и автономную доставку пяти медицинских грузов с использованием системы технического зрения.

Формат входных данных

Симулятор полета БЛА; навигационные параметры БЛА в текущий момент времени; видеопоток данных с бортовой камеры БЛА.

Формат выходных данных

Программа на Python, реализующая автоматический расчет координат местоположений точек доставки груза с использованием алгоритмов технического зрения, а также собственно автоматическую доставку грузов.

Критерии оценивания

- Оценивается факт доставки груза в каждую из пяти точек. Доставка считается успешной при попадании груза в обозначенную область радиусом 5 м.
- Время полета БЛА ограничено 6 мин.
- За каждую успешную доставку начисляется 3 балла. При этом решение участников должно пройти квалификационный тест.
- При провале квалификационного теста за каждую успешную доставку начисляется 1 балл.
- При успешной доставке всех пяти грузов начисляются дополнительные 2 балла.
- Правильный ответ на вопрос по теории прибавляет 1 балл (всего 2 вопроса).

Решение

Вариант алгоритма решения задачи:

1. Сформировать траекторию полета БЛА; написать код алгоритма траекторного управления для реализации этой траектории.
2. Отладить алгоритм определения координат точки доставки груза.
3. Продумать логику сброса груза в труднодоступных областях.
4. Разработать алгоритм обработки навигационных данных при наличии отказов измерительных систем.
5. Отработать разработанные алгоритмы на симуляторе.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```
1 # -*- coding: utf-8 -*-
2 import socket
```

```

3  from threading import Thread
4  import time
5  from math import atan2, cos, sin, sqrt, pi, atan
6  import cv2
7  import numpy as np
8  import uav_state
9  import serial
10
11 import subprocess
12
13 # Последние полученные данные телеметрии
14 last_telemetry = uav_state.uav_state()
15 running = True
16 capture = cv2.VideoCapture()
17
18 ser = serial.Serial(
19     port='COM6',
20     baudrate = 115200,
21     parity = serial.PARITY_NONE,
22     stopbits = serial.STOPBITS_ONE,
23     bytesize = serial.EIGHTBITS,
24     timeout = 0.1 )
25
26 # UDP сервер для параллельного получения данных телеметрии
27 def udp_server_thread():
28     host = '127.0.0.1'
29     port = 63504
30     addr = (host, port)
31     server = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
32     server.bind(addr)
33
34     while running:
35         d = server.recvfrom(1024)
36         received = d[0]
37         addr = d[1]
38
39         received = received.decode("utf-8")
40         received = received.replace("\t\r\n", "")
41         #print("---")
42         #print(received)
43         arr = [float(i) for i in received.split('\t')]
44         if len(arr) == 18:
45             last_telemetry.update(arr)
46     server.close()
47
48 def sendNewWaypoint(newx, newz, newh):
49     ser.write("wpnt {:.2f} {:.2f} {:.2f} \n".format(newx, newz, newh).encode())
50     print("wpnt {:.2f} {:.2f} {:.2f}".format(newx, newz, newh))
51
52 def sendDropCargo():
53     ser.write("drop \n".encode())
54     print("cargo drop attempt!")
55
56 # УЧАСТНИКАМ НТО - ЗАДАЧИ 6,7,8 -----
57 def find_surroundings(image, center, radius):
58     dots = []
59     h, w = image.shape
60     for row in range(center[1] - radius, center[1] + radius):
61         if row < h and row >= 0:
62             dots.append(image[row][center[0]])

```

```

63     for col in range(center[0] - radius, center[0] + radius):
64         if col < w and col >= 0:
65             dots.append(image[center[1]][col])
66
67     avg_val = sum(dots) / len(dots)
68     return avg_val
69
70     # Поток обработки данных с камеры
71     def camera_processing_thread():
72         # Параметры камеры
73         F = 6.1 * 1e-3 # Фокусное расстояние
74         dv = 7.8 * 1e-6 # высота пикселя
75         dh = 9.2 * 1e-6 # ширина пикселя
76         xadj = 2
77         zadj = -8
78         cap = cv2.VideoCapture("rtp-forwarder.sdp")
79         #cap = cv2.VideoCapture("circle.png")
80         cap.set(cv2.CAP_PROP_BUFFERSIZE, 0)
81
82         yellow_lower = np.array([20, 100, 100])
83         yellow_upper = np.array([30, 255, 255])
84
85         print("ready to search")
86
87         coords = (200, 900)
88         stage = 1 # wp number
89         evasion = False
90         drop = False
91         point_found = False
92         sendNewWaypoint(-245, 40, 80)
93         target = (-245, 40)
94         drop_dist = 0
95         # Основной цикл программы
96         while running:
97             # Получение изображения с бортовой камеры
98             ret, frame = cap.read()
99             point_found = False
100            tele = last_telemetry
101            X = tele.x
102            Z = tele.z - 8
103            H = tele.h
104            Heading = tele.yaw * pi / 180
105            dist = sqrt( (X - target[0])**2 + (Z - target[1])**2)
106
107            if ret and not evasion and not drop:
108                height, width, channels = frame.shape
109                cv2.normalize(frame, frame, 0, 255, cv2.NORM_MINMAX)
110                frame = cv2.GaussianBlur(frame, (7, 7), 0)
111                hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
112                yellow_mask = cv2.inRange(hsv, yellow_lower, yellow_upper)
113                full_mask = yellow_mask
114                contours, hierarchy = cv2.findContours(full_mask, cv2.RETR_LIST,
115                ↪ cv2.CHAIN_APPROX_SIMPLE)
116                maxlen = 0
117                if len(contours) > 0:
118                    truecont = contours[0]
119                    for icontour in contours:
120                        if len(icontour) > maxlen:
121                            maxlen = len(icontour)
122                            truecont = icontour

```

```

122     # # print(f"max green contour: {maxlen}")
123     cv2.drawContours(frame, truecont, -1, (0,0,255), 2)
124     sum_x = 0
125     sum_y = 0
126     for i in range(0, len(truecont)):
127         sum_x += truecont[i][0][0]
128         sum_y += truecont[i][0][1]
129     avg_x = int(round(sum_x / len(truecont)))
130     avg_y = int(round(sum_y / len(truecont)))
131
132     #print("Found the car on the parking lot : {} hor, {} ver pixels
133     ↪ from the corner".format(avg_x, avg_y))
134     cv2.ellipse(frame, (avg_x, avg_y), (5, 5), 0, 0, 360, (255, 255,
135     ↪ 255))
136     dxp = avg_x - width / 2
137     dyp = height/2 - avg_y
138     dxx = H / F * dh
139     dxy = H / F * dv
140     # print(f"Pixel size in meters: {dxx}")
141     shift_x = dxp * dxx
142     shift_y = dyp * dxy
143     # print("Contour center location: {} hor, {} ver pixels from the
144     ↪ center".format(dxp, dyp))
145     # print("Contour center location: {} hor, {} ver meters from the
146     ↪ center".format(shift_x, shift_y))
147     geo_x = shift_y * cos(Heading) - shift_x * sin(Heading) + X
148     geo_y = shift_y * sin(Heading) + shift_x * cos(Heading) + Z
149     #print("UAV coords X:{:2f} Z:{:2f} Heading:{}".format(X, Z,
150     ↪ Heading))
151
152     appr = sqrt((geo_x - coords[0]) ** 2 + (geo_y - coords[1]) ** 2)
153     if appr > 250:
154         print("Delivery coords X:{:2f} Z:{:2f}".format(geo_x, geo_y))
155         coords = (geo_x, geo_y)
156         point_found = True
157     if dist < 15:
158         print("recognition failed :(")
159
160     if point_found:
161         print(f"Evasion stage: {stage}")
162         if stage == 1:
163             target = (-400, 50)
164             sendNewWaypoint(target[0], target[1], 70)
165             evasion = True
166         if stage == 2:
167             target = (-580, -700)
168             sendNewWaypoint(target[0], target[1], 75)
169             evasion = True
170         if stage == 3:
171             target = (350, -590)
172             sendNewWaypoint(target[0], target[1], 70)
173             evasion = True
174         if stage == 4:
175             target = (750, -500)
176             sendNewWaypoint(target[0], target[1], 75)
177             evasion = True
178         if stage == 5:
179             target = (520, 450)
180             sendNewWaypoint(target[0], target[1], 75)
181             evasion = True

```

```

177
178     dist = sqrt((X - target[0]) ** 2 + (Z - target[1]) ** 2)
179     if evasion and dist < 30:
180         evasion = False
181         drop = True
182         if stage == 1:
183             target = (coords[0] - 4, coords[1] + 3)
184             drop_dist = 81
185         if stage == 2:
186             target = (coords[0] + 3, coords[1] + 28)
187             drop_dist = 40
188         if stage == 3:
189             target = (coords[0] + 5, coords[1] + 0)
190             drop_dist = 85
191         if stage == 4:
192             target = (coords[0] - 2, coords[1] - 2)
193             drop_dist = 73
194         if stage == 5:
195             target = (coords[0] - 0, coords[1] - 0)
196             drop_dist = 75
197         print(f"Approaching to drop. stage: {stage}")
198         sendNewWaypoint(target[0], target[1], 59)
199     dist = sqrt((X - target[0]) ** 2 + (Z - target[1]) ** 2)
200     if drop and dist < drop_dist:
201         drop = False
202         sendDropCargo()
203         stage = stage + 1
204         if stage == 2:
205             target = (-455, -770)
206             sendNewWaypoint(target[0], target[1], 80)
207         if stage == 3:
208             target = (300, -745)
209             sendNewWaypoint(target[0], target[1], 80)
210         if stage == 4:
211             target = (910, -670)
212             sendNewWaypoint(target[0], target[1], 80)
213         if stage == 5:
214             target = (650, 445)
215             sendNewWaypoint(target[0], target[1], 80)
216
217         #cv2.imshow("computer vision", frame)
218         #cv2.waitKey(3)
219         time.sleep(0.005)
220
221
222     # Основная функция для управления потоками
223     if __name__ == '__main__':
224         # Open RTP livestream through golang script
225         sendNewWaypoint(-245, 40, 80)
226         subprocess.Popen(["go", "run", "./main.go"])
227         time.sleep(5)
228         capture_thr = Thread(target=udp_server_thread, args=[])
229         capture_thr.daemon = True
230         capture_thr.start()
231         process_thr = Thread(target=camera_processing_thread, args=[])
232         process_thr.daemon = True
233         process_thr.start()
234
235     try:
236         while running:

```

```
237         time.sleep(1)
238     except KeyboardInterrupt:
239         print('\ninterrupted!')
```

Материалы для подготовки

- Библиотека OpenCV <https://stepik.org/course/116539/promo>.

Критерии определения победителей и призеров

Первый отборочный этап

В первом отборочном этапе участники решали задачи предметного тура по двум предметам: физике и информатике и инженерного тура. В каждом предмете максимально можно было набрать 100 баллов, в инженерном туре 100 баллов. Для того, чтобы пройти во второй этап участники должны были набрать в сумме по обоим предметам не менее 50 баллов, независимо от уровня.

Второй отборочный этап

Количество баллов, набранных при решении всех задач второго отборочного этапа, суммируется. Победители второго отборочного этапа должны были набрать не менее 73 балла, независимо от уровня.

Заключительный этап

Индивидуальный предметный тур

- физика — максимально возможный балл за все задачи — 100 баллов;
- информатика — максимально возможный балл за все задачи — 100 баллов.

Командный инженерный тур

Команды заключительного этапа получали за командный инженерный тур от 0 до 100 баллов: команда, набравшая наибольшее число баллов среди других команд, становилась командой-победителем.

Все результаты команд нормировались по формуле:

$$\frac{100 \times x}{MAX},$$

где x — число баллов, набранных командой,

MAX — число баллов, максимально возможное за инженерный тур.

В заключительном этапе олимпиады индивидуальные баллы участника складываются из двух частей, каждая из которых имеет собственный вес: баллы за индивидуальное решение задач по предметам (физика, информатика) с весом $K_1 = 0,15$ каждый предмет и баллы за командное решение задач инженерного тура с весом $K_2 = 0,7$.

Итоговый балл определяется по формуле:

$$S = K_1 \cdot (S_1 + S_2) + K_2 \cdot S_3,$$

где S_1 — балл первой части заключительного этапа по физике (предметный тур) в стобальной системе ($S_{1 \text{ макс}} = 100$);

S_2 — балл первой части заключительного этапа по информатике (предметный тур) в стобальной системе ($S_{2 \text{ макс}} = 100$);

S_3 — итоговый балл инженерного командного тура в стобальной системе ($S_{3 \text{ макс}} = 100$).

Итого максимально возможный индивидуальный балл участника заключительного этапа = 100 баллов.

Критерий определения победителей и призеров

Чтобы определить победителей и призеров (независимо от класса) на основе индивидуальных результатов участников, был сформирован общий рейтинг всех участников заключительного этапа. С начала рейтинга были выбраны 3 победителя и 6 призеров (первые 25% участников рейтинга становятся победителями или призерами, из них первые 8% становятся победителями, оставшиеся — призерами).

Критерий определения победителей и призеров (независимо от уровня)

Категория	Количество баллов
Победители	77,90 и выше
Призеры	От 56,10 до 75,50

Работа наставника после НТО

Участие школьника в Олимпиаде может завершиться после любого из этапов: первого или второго отборочных либо после заключительного этапа. В каждом случае после завершения участия наставнику необходимо провести с учениками рефлексию — обсудить полученный опыт и проанализировать, что позволило достичь успеха, а что привело к неудаче.

Важная задача наставника — превратить неудачу в инструмент будущего успеха. Для этого необходимо вместе с учениками наметить план развития компетенций и подготовки к будущему сезону Олимпиады. Подробные материалы о проведении рефлексии представлены в курсе «Наставник НТО»: <https://academy.sk.ru/events/310>.



Наставнику важно проинформировать руководство образовательного учреждения, если его учащиеся стали финалистами, призерами и победителями. Публичное признание высоких результатов дополнительно повышает мотивацию.

В процессе рефлексии с учениками, не ставшими призерами или победителями, рекомендуется уделить особое внимание особенностям командной работы: распределению ролей, планированию работы, возникающим проблемам. Для этого могут использоваться опросники для самооценки собственной работы и взаимной оценки участниками других членов команды (P2P). Такие опросники могут выявить внутренние проблемы команды, для решения которых в план подготовки можно добавить мероприятия, направленные на ее сплочение.

Стоит рассказать, что в истории НТО было много примеров, когда не победив в первый раз, на следующий год участники показывали впечатляющие результаты, одержав победу сразу в нескольких профилях. Конечно, важно отметить, что так происходит только при учете прошлых ошибок и подготовке к Олимпиаде в течение года.

Еще одним направлением работы наставника после НТО может стать создание кружка по направлению профилей или по формированию необходимых компетенций: программирование, электроника, робототехника, 3D-моделирование и т. п. Формат подобного кружка может быть различным: короткие модули, дополнительные курсы, факультативы, группы дополнительного образования. Для создания кружков можно воспользоваться образовательными программами, опубликованными на сайте НТО: <https://ntcontest.ru/mentors/education-programs/>.



Важным фактором успешного участия в следующих сезонах НТО может стать поддержка родителей учеников. Знакомство с родителями помогает наставнику продемонстрировать им важность компетенций, развиваемых в процессе участия в НТО, для будущего образования и карьеры школьников. Поддержка родителей помогает мотивировать участников и позволяет выделить необходимое время на занятия в кружке.

С участниками-выпускниками наставнику рекомендуется обсудить их дальнейшее профессиональное развитие и его связь с выбранными профилями НТО. Отдельно можно обратить внимание на льготы для победителей и призеров, предлагаемые в вузах с интересующими ученика направлениями. Кроме того, ряд вузов предлагает льготы для всех финалистов НТО, а также учитывает результаты Конкурса цифровых портфолио «Талант НТО».

